Neural Networks for Document Image and Text Processing

Joan Pastor Pellicer

Supervisor: Dra. María José Castro Bleda

September 2017

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

A l'Oncle.

Acknowledgements

After several years working on this Ph.D. and several internships is difficult to recall all the partners, colleagues, family and friends that have accompanied me during this trip; I hope I haven't forgotten anybody.

In the first place, I would like to thank my parents, I have been so lucky of having their support all the time. They never questioned any of my decisions and helped without hesitating. They got as excited as me when I joined this Ph.D. and finally, I could complete it. I don't want to forget my sister, whose support has been very valuable, she has been a pillar, especially during the bad moments, and she has been always there. I wanted to thank the rest of my family who has been also very interested and happy that I took this path.

I could not have gotten far on this Ph.D. if it weren't by my supervisor Maria Jose; her guidance has been vital. And, of course, my other colleagues: Pako and Salva. This work is taken from their seed and I continued what they started. Also, I appreciate the ELiRF group at UPV that hosted me during all this time and made me feel one of them from the very first day.

To my colleagues and all the people which I shared the lab with: Marcos, Ximo, Jorge, Lucia, Merche, Santi, Merc, Parth, Soren, Maite and Victor. It has been a pleasure to sit next to you and learning a lot from you, you made me grow as a researcher. Next, I would like to thank my friends and also Ph.D. fellows at the university (GH): Flores, Jaume, Miquel, Joan, and Mario. I had a lot of fun with the long coffee chats after lunch, parties, and push-ups, you made my time in València.

My first internship was at Fribourg and I have to admit that it was one of the best experiences of my life, and that was thanks to the awesome team at DIVA Group: Markus, Angie, Matias, Mattias, Marcel, Fotini, Rolf, and also to all the friends that I made there. Later, I had a similar experience during my stay

at Kaiserslautern where I enjoyed my time there with awesome colleagues: Federico, Shan, Adnan, Wonmin and friends with whom I enjoyed my time with. My last internship was on the Handwriting team at Google. I had an awesome experience and the luck of working on topics related to this work. During the stay, I grow up as an engineer and all thanks to the confidence and friendship of Thomas, Victor, Pedro, Sandro, Ashok, Marcos, Philippe, Lilun, Yasuhisa, and Dmitri.

I do not want to forget my unconditional friends from Gandia: Carlos, Elena, Vicen, Tamara, Ximo and Mireia who accompanied me during this journey.

And last but not the least, I have to thank the person that has insisted the most about my thesis, asking (and sometimes inconveniently) about it, trying to decipher what the work is about and pushing me to finish it. He has been the first doctor of my family and the proudest of the following work. Oncle, wherever you are: “Ho he aconseguit!”

Abstract

Nowadays, the main libraries and document archives are investing a considerable effort on digitizing their collections. Indeed, most of them are scanning the documents and publishing the resulting images without their corresponding transcriptions. This seriously limits the document exploitation possibilities. When the transcription is necessary, it is manually performed by human experts, which is a very expensive and error-prone task. Obtaining transcriptions to the level of required quality demands the intervention of human experts to review and correct the resulting output of the recognition engines. To this end, it is extremely useful to provide interactive tools to obtain and edit the transcription.

Although text recognition is the final goal, several previous steps (known as preprocessing) are necessary in order to get a fine transcription from a digitized image. Document cleaning, enhancement, and binarization (if they are needed) are the first stages of the recognition pipeline. Historical Handwritten Documents, in addition, show several degradations, stains, ink-trough and other artifacts. Therefore, more sophisticated and elaborate methods are required when dealing with these kinds of documents, even expert supervision in some cases is needed. Once images have been cleaned, main zones of the image have to be detected: those that contain text and other parts such as images, decorations, versal letters. Moreover, the relations among them and the final text have to be detected. Those preprocessing steps are critical for the final performance of the system since an error at this point will be propagated during the rest of the transcription process.

The ultimate goal of the Document Image Analysis pipeline is to receive the transcription of the text (Optical Character Recognition and Handwritten Text Recognition). During this PhD Thesis we aimed to improve the main stages of the recognition pipeline, from the scanned documents as input to the final transcription. We focused our effort on applying Neural Networks and deep

learning techniques directly on the document images to extract suitable features that will be used by the different tasks dealt during the following work: Image Cleaning and Enhancement (Document Image Binarization), Layout Extraction, Text Line Extraction, Text Line Normalization and finally decoding (or text line recognition). As one can see, the following work focuses on small improvements through the several Document Image Analysis stages, but also deals with some of the real challenges: historical manuscripts and documents without clear layouts or very degraded documents.

Neural Networks are a central topic for the whole work collected in this document. Different convolutional models have been applied for document image cleaning and enhancement. Connectionist models have been used, as well, for text line extraction: first, for detecting interest points and combining them in text segments and, finally, extracting the lines by means of aggregation techniques; and second, for pixel labeling to extract the main body area of the text and then the limits of the lines. For text line preprocessing, i.e., to normalize the text lines before recognizing them, similar models have been used to detect the main body area and then to height-normalize the images giving more importance to the central area of the text. Finally, Convolutional Neural Networks and deep multilayer perceptrons have been combined with hidden Markov models to improve our transcription engine significantly.

The suitability of all these approaches has been tested with different corpora for any of the stages dealt, giving competitive results for most of the methodologies presented.

Resumen

Hoy en día, las principales librerías y archivos está invirtiendo un esfuerzo considerable en la digitalización de sus colecciones. De hecho, la mayoría están escaneando estos documentos y publicando únicamente las imágenes sin transcripciones, limitando seriamente la posibilidad de explotar estos documentos. Cuando la transcripción es necesaria, esta se realiza normalmente por expertos de forma manual, lo cual es una tarea costosa y propensa a errores. Si se utilizan sistemas de reconocimiento automático se necesita la intervención de expertos humanos para revisar y corregir la salida de estos motores de reconocimiento. Por ello, es extremadamente útil para proporcionar herramientas interactivas con el fin de generar y corregir la transcripciones.

Aunque el reconocimiento de texto es el objetivo final del Análisis y Procesamiento de Documentos, varios pasos previos (conocidos como preprocesamiento) son necesarios para conseguir una buena transcripción a partir de una imagen digitalizada. La limpieza, mejora y binarización de las imágenes (si son necesarios) son las primeras etapas del proceso de reconocimiento. Además, los manuscritos históricos tienen una mayor dificultad en el preprocesamiento, puesto que pueden mostrar varios tipos de degradaciones, manchas, tinta a través del papel y demás dificultades. Por lo tanto, este tipo de documentos requiere métodos de preprocesamiento más sofisticados. En algunos casos, incluso, se precisa de la supervisión de expertos para garantizar buenos resultados en esta etapa. Una vez que las imágenes han sido limpiadas, las diferentes zonas de la imagen deben de ser localizadas: texto, gráficos o dibujos, decoraciones, letras versales, etc. Por otra parte, también es importante conocer las relaciones entre estas entidades y el texto. Estas etapas del preprocesamiento son críticas para el rendimiento final del sistema, ya que los errores cometidos en aquí se propagarán al resto del proceso de transcripción.

El objetivo principal del trabajo presentado en este documento es mejorar las principales etapas del proceso de reconocimiento completo: desde las imágenes escaneadas hasta la transcripción final. Nuestros esfuerzos se centran en aplicar técnicas de Redes Neuronales y aprendizaje profundo directamente sobre las imágenes de los documentos, con la intención de extraer características adecuadas para las diferentes tareas: Limpieza y Mejora de Documentos, Extracción de Líneas, Normalización de Líneas de Texto y, finalmente, transcripción del texto. Como se puede apreciar, el trabajo se centra en pequeñas mejoras en diferentes etapas del Análisis y Procesamiento de Documentos, pero también trata de abordar tareas más complejas: manuscritos históricos, o documentos que presentan serias degradaciones.

Las redes neuronales y el aprendizaje profundo son uno de los temas centrales de esta tesis. Diferentes modelos neuronales convolucionales se han desarrollado para la limpieza y mejora de imágenes de documentos. También se han utilizado modelos conexionistas para la tarea de extracción de líneas: primero, para detectar puntos de interés y segmentos de texto y, agregarlos para extraer las líneas del documento; y en segundo lugar, etiquetando directamente los píxeles de la imagen para extraer la zona central del texto y así definir los límites de las líneas. Para el preproceso de las líneas de texto, es decir, la normalización del texto antes del reconocimiento final, se han utilizado modelos similares a los mencionados para detectar la zona central del texto. Las imagenes se rescalan a una altura fija dando más importancia a esta zona central. Por último, en cuanto a reconocimiento de escritura manuscrita, se han combinado técnicas de redes neuronales y aprendizaje profundo con Modelos Ocultos de Markov, mejorando significativamente los resultados obtenidos previamente por nuestro motor de reconocimiento.

La idoneidad de todos estos enfoques han sido testeados con diferentes corpus en cada una de las tareas tratadas., obteniendo resultados competitivos en la mayor parte de los casos analizados.

Resum

Avui en dia, les principals llibreries i arxius històrics estan invertint un esforç considerable en la digitalització de les seues col·leccions de documents. De fet, la majoria estan escanejant aquests documents i publicant únicament les imatges sense les seues transcripcions, fet que limita seriosament la possibilitat d'explotació d'aquests documents. Quan la transcripció del text és necessària, normalment aquesta és realitzada per experts de forma manual, la qual cosa és una tasca costosa i pot provocar errors. Si s'utilitzen sistemes de reconeixement automàtic es necessita la intervenció d'experts humans per a revisar i corregir l'eixida d'aquests motors de reconeixement. Per aquest motiu, és extremadament útil proporcionar eines interactives amb la finalitat de generar i corregir les transcripcions generades pels motors de reconeixement.

Tot i que el reconeixement del text és l'objectiu final de l'Anàlisi i Processament de Documents, diversos passos previs (coneguts com preprocessament) són necessaris per a l'obtenció de transcripcions acurades a partir d'una imatge digitalitzada. La neteja, millora i binarització de les imatges (si calen) són les primeres etapes prèvies al reconeixement. A més a més, els manuscrits històrics presenten una major dificultat d'analisi i preprocessament, perquè poden mostrar diversos tipus de degradacions, taques, tinta a través del paper i altres peculiaritats. Per tant, aquest tipus de documents requereixen mètodes de preprocessament més sofisticats. En alguns casos, fins i tot, es precisa de la supervisió d'experts per a garantir bons resultats en aquesta etapa. Una vegada que les imatges han sigut netejades, les diferents zones de la imatge han de ser localitzades: text, gràfics o dibuixos, decoracions, versals, etc. D'altra banda, també és important conéixer les relacions entre aquestes entitats i el text que contenen. Aquestes etapes del preprocessament són crítiques per al rendiment final del sistema, ja que els errors comesos en aquest moment es propagaran a la resta del procés de transcripció.

L'objectiu principal del treball que estem presentant és millorar les principals etapes del procés de reconeixement, és a dir, des de les imatges escanejades fins a l'obtenció final de la transcripció del text. Els nostres esforços se centren en aplicar tècniques de Xarxes Neuronals i aprenentatge profund directament sobre les imatges de documents, amb la intenció d'extraure característiques adequades per a les diferents tasques analitzades: neteja i millora de documents, extracció de línies, normalització de línies de text i, finalment, transcripció del text. Com es pot apreciar, el treball realitzat aplica xicotetes millores en diferents etapes de l'Anàlisi i Processament de Documents, però també tracta d'abordar tasques més complexes: manuscrits històrics, o documents que presenten serioses degradacions.

Les xarxes neuronals i l'aprenentatge profund són un dels temes centrals d'aquesta tesi. Diferents models neuronals convolucionals s'han desenvolupat per a la neteja i millora de les imatges dels documents. També s'han utilitzat models connexionistes per a la tasca d'extracció de línies: primer, per a detectar punts d'interés i segments de text i, agregar-los per a extraure les línies del document; i en segon lloc, etiquetant directament els pixels de la imatge per a extraure la zona central del text i així definir els límits de les línies. Per al preprocés de les línies de text, és a dir, la normalització del text abans del reconeixement final, s'han utilitzat models similars als utilitzats per a l'extracció de línies. Finalment, quant al reconeixement d'escriptura manuscrita, s'han combinat tècniques de xarxes neuronals i aprenentatge profund amb Models Ocults de Markov, que han millorat significativament els resultats obtinguts prèviament pel nostre motor de reconeixement.

La idoneïtat de tots aquests enfocaments han sigut testejats amb diferents corpus en cadascuna de les tasques tractades, obtenint resultats competitius en la major part dels casos analitzats.

Contents

List of Figures

List of Tables

List of Acronyms

BDLSTM Bidirectional Long Short Term Memories

BDRNN Bidirectional Recurrent Neural Network

BP Backpropagation

BTT Backpropagation-through-time

CC Connected Component

CER Character Error Rate

CE Cross-Entropy

Cnn+Mlp Combined CNN+MLP

CNN Convolutional Neural Network

CRF Conditional Random Fields

CTC Connectionist Temporal Classification

DBN Deep Belief Networks

DIA Document Image Analysis

DIBCO Document Image Binarization Contest

DIB Document Image Binarization

DoG Difference-of-Gaussian

DP Dynamic Programming

ELiRF The Natural Language Engineering and Pattern Recognition Group

EM Expectation-Maximization

FM F-Measure

GMM Gaussian mixture model

GPU Graphic Processor Unit

H-DIBCO Handwritten Document Image Binarization Contest

HMM/ANN Hidden Markov Models hybridized with ANNs

HMM/CNN Hidden Markov Models hybridized with CNNs

HMM/GMM Hidden Markov Models with Gaussian Mixture Models

HMM/RNN Hidden Markov Models with Recurrent Neural Networks

HMM Hidden Markov Model

HWR Handwriting Text Recognition

IIF International Image Interoperability

IP Interest Point

JSON JavaScript Object Notation

k-NN k-Nearest Neighbors

LEP Local Extrema Point

LM Language Model

LSTM Long Short Term Memories

LVCSR Large Vocabulary Continuous Speech Recognition

MBA Main Body Area

MDLSTM MultiDirectional Long Short Term Memories

MDRNN MultiDirectional Recurrent Neural Network

MERT Minimum Error Rate Training

MLP Multilayer Perceptron

ML Machine Learning

MRF Markov Random Fields

MSE Mean Squared Error

MSI MultiSpectral Images

MS Match Score

NNLM Neural Network Language Model

ANN Artificial Neural Network

OCR Optical Character Recognition

OOV Out-of-Vocabulary

PCA Principal Component Analysis

pdf Probability Density Function

PHR Pixel Level Hit Rate

PSNR Peak Signal to Noise Ratio

RBM Restricted Boltzmann Machine

ReLU Rectified Linear Unit

RLSA Run-Length Smearing Algorithm

RNN Recurrent Neural Network

ROVER Recognition Output Voting Error Reduction

SDAE Stacked Denoising Autoencoders

SER Sentence Error Rate

SIFT Scale Invariant Feature Transform

SURF Speeded Up Robust Features

SVM Support Vector Machine

TEI Text Encode Initiative

Tiff Tagged Image File Format

Tla Text Line Accuracy

TLE Text Line Extraction

TLL-DR Text-Line-Level Detection Rate

TLN Text Line Normalization

TL-PA Text Line Pixel Accuracy

WER Word Error Rate

Chapter 1

Introduction

Contents

This introductory chapter prepares the reader to understand the overall work presented in this document. Interest in digitizing documents has increased with the proliferation of cheap storage devices and new digital document formats that allow documents to be stored and retrieved. All of these advances have made it easier and more convenient to keep digitized libraries instead of keeping physical volumes of data. We will establish the broad outlines followed in this work, and we will present the motivation and objectives as well as a brief summary of the structure of this document.

1.1 Motivation

The main libraries and document archives are currently digitizing their collections. The advantages of digitizing documents are evident: documents can be universally accessed; several people can access the same resources at the same time; in some cases, indexing and searching for content can be performed. And, of course, it saves paper and does not require physical space. Most libraries and archives are scanning the documents and publishing the

resulting images without their corresponding transcriptions. This seriously limits the possibilities of document exploitation. Document Image Analysis aims to analyze and extract useful information from scanned documents. It involves all of techniques and algorithms that are applied to these document images to obtain digital information related to them.

The most evident and ultimate objective of Document Image Analysis is to transcribe the content of the documents. Once a reliable transcription is obtained, this information allows the contents to be searched and indexed, and allows more meaningful information to be extracted for better understanding. Initially, Document Image Analysis, (specifically Optical Character Recognition) was applied to automatic mail address recognition which saved a tremendous amount of effort to the U.S Postal service in the 60s. However, the scope of Document Image Analysis has grown over the years: form recognition, writer identification, signature verification, content search, queries, script recognition, meta-data identification (italic, bold, URLs, references, style, glyphs, symbols), document classification, reading order and much, much more. In this PhD Thesis, we aim to enhance both purposes of Document Image Analysis, which is getting the final transcription of a scanned document image, and the pre-processing stages, which include all of the transformations performed on the image before the final recognition.

Our journey starts with the legacy of the The Natural Language Engineering and Pattern Recognition Group. That team developed a Handwriting Text Recognition engine which achieved great results in modern handwritten texts (Offline IAM Database) [Espana-Boquera et al. 2011]. Part of their success was developing a bright text line pre-processing and the effort invested in making the line images suitable for the final recognition (decoding) stage. From these previous ideas, we realized that some of the techniques and insights could be applied to other pre-processing stages such as Text Line Extraction, layout analysis, or document cleaning and enhancement. For instance, an Artificial Neural Network-based image enhancement process was performed on text lines before text size normalization, but the target data was modern handwriting, which does not have high levels of degradation or distortions. Therefore, we decided to improve these connectionist models and apply them to full pages that included other artifacts such as degradations, decorations, borders, out-of-page zones, etc. In the process, we realized that most of the core techniques in the literature were powerful enough to deal with recent typewritten documents. However, there was still room for improvement, for instance, applying them to more complex documents such as historical documents or even modern unconstrained handwriting with complex layouts. We

also took the previous work based on Interest Point detection as a starting point. In previous works developed by the group for Text Line Normalization, Local Extrema Points were extracted and classified to track text reference lines. These ideas meant that Text Line Normalization could be used to detect text areas in the whole page, and, therefore, obtain lines as we will show later. Local Extrema Points are extracted by heuristics and then classified into Interest Points by means of Machine Learning techniques. An Artificial Neural Network receives contextual information around the point to be classified, and the output is one of the classes that gives information about the position of the point in the line. As stated, this method is extended to the whole page, where the Interest Points provide information about the text line. It is then feasible to join them to draw the line frontiers with a clustering algorithm.

Finally, the third encouraging pillar of this PhD Thesis comes from the success of deep learning techniques in Computer Vision-related tasks, specifically, the excellent performance obtained with Convolutional Neural Networks. There is an increasing trend towards directly using raw images without explicit feature extraction and then applying several layers of convolutions to extract edges and shapes, and combining them in higher hierarchies. This allows overcoming tasks such as image classification, object detection, image restoration, compression, etc. Documents are also images, and, even though they have different final objectives than typical scene pictures, most of the techniques developed for Computer Vision could be reused and adapted to Document Image Analysis methods.

1.2 Tasks and tools

In the field of Computer Science research, two tendencies can be found: tools and tasks. In short, tools are applied to tasks, and tasks require tools. Therefore, some works focus on providing a general solution to different problems, while others try to find the best option for the actual task. Besides, standard tasks (such as MNIST, RIMES, ImageNet, and others) can be used for both purposes.

In the case of tools, one develops a great technique and then tries to verify the greatness of its technology by trying it on different tasks. For instance, in the Machine Learning community, specifically with Artificial Neural Networks, the community develops faster and better methods to train, generalize, and achieve better rates. As soon as these new techniques become popular they start to be included in most of the Machine Learning frameworks and toolkits,

and the community will try them on their specific tasks. However, there is a dangerous drawback; sometimes one wants to apply his/her techniques to all possible tasks, regardless of whether or not they are the best or the right approach.

On the other hand, in the case of tasks, one wants to find the best way of performing them. We must probably rely on some techniques and ideas developed by others and adapt them to the given task. This approach could appear to be more straightforward since you only have to try several existing techniques and observe which one works better. However, there is nothing further from the truth. This approach involves a deep study of the task and significant engineering with all the pre- and post-processing in order to guarantee success.

Finally, we have the in-between situation: "I have this task, and I know this novel technique that will work for it." However, this also presents different challenges:

- Once you develop a novel technique, you and the whole community expect a significant improvement in the state-of-the-art methods. What is the point of applying this advanced technology if it cannot outperform the current models in the bibliography?
- Post- and pre-processing approaches could work well in some of the specific parts of the whole process, but there will be other stages that require transforming the data, applying the algorithm, and adapting the output to the desired answer of the system. And when using new methods, this is not straightforward and involves a significant amount of effort, research, and engineering.

Our research follows this last approach: we want to tackle the main Document Image Analysis stages. Indeed, we want to use some of the new advances on Artificial Neural Network and deep learning combined with our novel techniques to approach them.

1.3 End-to-end: The Handwriting Text Recognition pipeline

As stated, we have extended the scope of our research to the overall transcription process. One of the main achievements of this PhD Thesis is to provide support for (almost) the whole transcription process: take a handwritten document, clean it, extract the lines, and then transcribe them. Of course, some of the full pipeline stages have been overlooked, but in some cases those are optional. This is the case of skew correction or perspective correction in which one can only rely on external effective and proven methods that are included in several toolkits: OpenCV, OCROpus, Tesseract, and many others.

Therefore, we start with a scanned page. If it does not have a significant perspective distortion, we could directly apply the methods depicted in Chapter 4. We enhance the image by removing stains and noise artifacts and by extracting the primary foreground information (mostly text). Indeed, it is possible to eliminate areas like margins, outside of page, decorations and other parts that are not useful for the rest of the processing steps.

The next step involves extracting the lines to be transcribed. At this point, we start with the cleaned page, and any of the techniques introduced in Chapter 5 could be used for this purpose. In fact, since some of these methods work directly on the raw image, it is possible to train these Text Line Extraction techniques to deal with the expected noise. In other words, the cleaning step is implicitly included into the Text Line Extraction models. Nevertheless, splitting these stages into two steps (cleaning/enhancement and line segmentation) is also a valid approach. In addition, if the document structure is complex, it is desirable to separate the line extraction in two steps as well: layout analysis to extract text blocks and removal of other parts; and then line segmentation of the detected blocks.

With the extracted lines, we already have the data that will be fed into the recognizer, but first, we would like to normalize the lines. The objective here is twofold:

- To have images with a fixed height because our recognition engine expects a fixed size column height.
- To reduce variability between different samples and styles.

In this process, there are a few pre-processing steps that have been taken from previous text line pre-processing [Espana-Boquera et al. 2011]: slope, skew, and slant correction. However, the height normalization stage has been improved by classifying all the line pixels instead of only the Interest Point pixels. This thorough method gave us a significant improvement as explained in Chapter 6.

Finally, our Handwriting Text Recognition engine is applied to the normalized text line images. The optical modeling and, specifically, the feature extraction process have been improved and updated with deep learning techniques to learn more useful characteristics of the text. Hence, better and more accurate results are achieved. The full process is illustrated in Figure 1.1.

As we have seen, we deal with the main stages of Document Image Analysis, either by creating new ones from scratch or by improving the existing ones.

1.4 Objectives

After knowing the challenges of the Document Image Analysis and Optical Character Recognition, we present the main goals of this PhD Thesis.

1. To extend our recognition system to include more pre-processing stages. The goal is to cover the main steps of the Document Image Analysis pipeline: Cleaning and enhancement, and Layout and Text Line Extraction.

2. To create reliable Machine Learning-based models for document cleaning and Text Line Extraction that could be easily adapted to different documents and collections. Instead of having an adaptive or a generic method that provides a reasonable performance in different kinds of documents, we suggest investing a small amount of effort in supervising or adapting the already trained classifiers, which in the long term usage eases the overall transcription process.

3. To develop a specific Artificial Neural Network (ANN) training method able to deal with unbalanced datasets since most of the task analyzed suffer this issue. Focusing on the evaluated metrics i.e. F-Measure (FM).

4. To provide tools that assist these processes. For some of the techniques shown here, we require supervised data. For this purpose, we will pro-

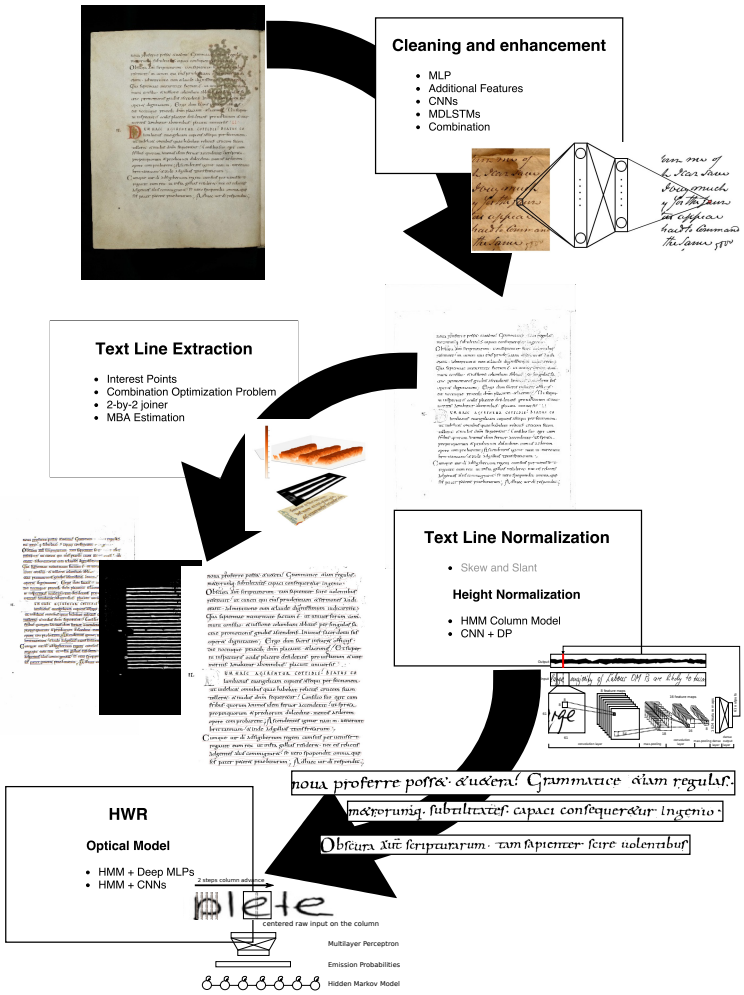


Figure 1.1: The figure illustrates the techniques and methods addressed in this PhD Thesis. First, the image is cleaned and enhanced. Then, methods for Text Line Extraction can be applied, and finally, the height of the image is normalized. In the last step, the recognition engine has been improved.

vide a new corpus and tools that support and facilitate the human supervision process.

5. To update our decoding engine with deep learning techniques. New approaches and improvements in recent years have been published for Handwriting Text Recognition. Most of them apply deep learning techniques, and those ideas can be included in our previously developed Handwriting Text Recognition engine. The goal is to adapt and implement these new advances to improve its performance.

6. To explore and adapt Computer Vision techniques to Document Image Analysis tasks. Most of the newer techniques in Computer Vision and Image Processing can also be applied to documents since the input is simply a scanned document.

We developed and implemented different ideas and methods to achieve the objectives mentioned above. Thus, we are encouraged to write reliable, clean, efficient and useful programs to allow the users to interact easily with them. Other technical objectives that are related to the implementation of our models are:

(a) Efficient implementation of the described algorithms for the different tasks.

(b) Contribution to the April-ANN [F. Zamora-Martínez, España-Boquera, et al. 2013] toolkit developed by the research team. Apart from the actual contribution to this toolkit during the development of this PhD Thesis, we intend to continue modifying and updating the general developed techniques.

1.5 Document structure

We briefly describe, in this section, how the current document has been structured, and we summarize the contents that can be found in the following chapters.

Chapter 2. Related work For better comprehension of the tasks and the current methodologies, we collected some of the useful resources to analyze the evolution and the techniques that have been applied to the above-mentioned tasks. The chapter is structured following the four main tasks in this PhD Thesis (indicating the most relevant techniques and trends in each of them): Document Image Binarization, Text Line Extraction, Text Line Normalization, and Decoding.

Chapter 3. Artificial Neural Networks Artificial Neural Networks are a central topic to the whole work collected in this document. In this introductory chapter, basic concepts about Artificial Neural Networks and the other statistical models used in the rest of the chapters are introduced. We also discuss some of the techniques and tricks used to speed up convergence and to avoid over-fitting.

Chapter 4. Document Image Cleaning and Enhancement Document cleaning and enhancement is one of the first stages of the Handwriting Text Recognition pipeline. It works directly on the scanned image, and the errors and mistakes commited there will affect the rest of the process. In this chapter, we discuss several techniques based on Artificial Neural Networks. First, we review the traditional Multilayer Perceptron used in previous work; then we improve it by adding new features. Two novel techniques that are applied in this field are presented: Convolutional Neural Networks and MultiDirectional Long Short Term Memories. We introduce and discuss the different corpora and metrics used to evaluate this task and the results of the proposed approaches.

Chapter 5. Text Line Extraction Several approaches for Text Line Extraction are presented and evaluated in this chapter. On the one hand, we focus on the use of Interest Points to extract text segments and then combine them in a second stage that aggregates the different segments to remove the final lines. On the other hand, pixel labeling is applied to obtain the Main Body Area in the raw images and to finally segment the generated map in order to define the line frontiers. In addition, we discuss the different metrics and formats contributed by several authors and their suitability for this task.

Chapter 6. Text Line Normalization This chapter works directly with the text line to be transcribed. In the original work, the three principal zones of the text lines are extracted: Ascenders, the Main Body Area, and Descenders. Then they are height-normalized independently. In this PhD Thesis, we have improved the previous text height normalization techniques based on Interest Points but classifying the pixel of the whole image. First, Hidden Markov Models are applied to constrict the zones of each column. In a later approach, we present a model that is based on Dynamic Programming and Convolutional Neural Networks that solve the problems derived from the first approach.

Chapter 7. Decoding from Scratch In this chapter, we focus on the decoding stage of Handwriting Text Recognition. We directly use text line images by applying Convolutional Neural Networks and deep Multilayer Perceptrons for feature extraction in the Handwriting Text Recognition system. In addition, a qualitative study on the features learned by the Convolutional Neural Networks is performed to get a better understanding of the system behavior.

Chapter 8. F-Measure as Artificial Neural Network optimization function Imbalance datasets impose serious problems in Machine Learning. For many tasks characterized by imbalanced data, the F-Measure is commonly used when discussing the results and comparing them with other approaches. The chapter studies the use of F-Measure as a loss function for training ANN through the Backpropagation algorithm. This novel training criterion has been employed when training ANN for document cleaning and enhancements.

Chapter 9. Conclusions and Future Work In this chapter, we summarize the main discussions derived for each of the main tasks, and we also present a general overview of this PhD Thesis development. Finally, future lines of work and extensions are presented. In addition, each chapter includes its own summary and conclusions.

The reader will notice that most of the Chapters are self-contained, specifically Chapters 4, 5, 6 and 7, which treat different stages of the overall Document Image Analysis pipeline. Same thing with Chapter 8, which is self-contained. The structure of the chapters is typical: we introduce the specific task, the methodologies, the results, and a brief discussion. In addition, we include the index, introduction, and summary within the chapters. In any case, we encourage the reader to look at Chapter 3 in order to have a better overview of the common techniques and configurations that are shared in the different tasks.

Chapter 2

Related work

Contents

In this PhD Thesis many different stages of the Document Image Analysis pipeline are adressed: Document Image Binarization, Text Line Extraction and Text Line Normalization for document image processing and, in Chapter 7, we focus on the recognition. In this chapter we review the related work, we discuss the main approaches for Document Image Analysis and the most successful systems in order to have a better understanding of the presented tasks and their associated issues. It is not surprising, indeed, that most of the pioneer works in the researched topics worked relatively well and their ideas are still used and improved through time.

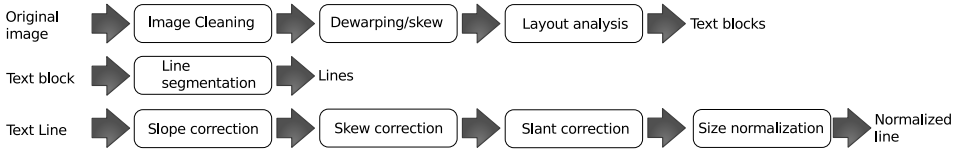


Figure 2.1: Preprocessing steps. The diagram shows the natural preprocessing workflow, from the original image to the lines, and then he previous transformation of the lines before the OCR process (Figure extracted and adapted from [España Boquera 2016]).

.

2.1 Document Image Binarization

Document Image Binarization (DIB) is one of the first stages of the Document Image Analysis (DIA) and the further recognition pipeline. It consists mainly in classifying the pixels of the image into background and foreground pixels. At this stage, "binarization" will imply some cleaning and enhancement, since detecting the foreground pixel will remove, for instance, ink spots, degradations or ink-bleed through, besides it will recover lost strokes or other foreground information. Undoubtedly, the performance of this stage will significantly affect the outcome of all the other document analysis steps like layout analysis and Text Line Extraction (TLE), and the final text transcription as well [Nagy 2000].

Binarization is also known in the literature as thresholding, since a threshold value is applied to the pixels to saturate them in one of the classes[1]. Image Binarization takes into account any images, but DIB is meant for documents where text (and other graphics) are considered foreground and the background is the "common background of the sheet".

In the literature we could find lots of works about Image Binarization (and of course, the DIB modality). While writing this section, the main problem was to find a proper taxonomy to classify the many related approaches. There are several ways of classification: technique used, type of the images they are aimed to, features used, etc. Indeed, any methodology could be described as a combination of these categories.

[1] In grayscale images, pixels are usually represented by 1 byte ($0 - 255$ range values), when binarizing we use the value 0 as background and 255 as foreground. These binary images can be easily represented by 1 bit: 0 or 1.

Moreover, the Document Image Binarization Contest (DIBCO) [B. Gatos et al. 2011; Pratikakis et al. 2011, 2013] and Handwritten Document Image Binarization Contest (H-DIBCO) [Ntirogiannis et al. 2014b; Pratikakis et al. 2012, 2010] have been held for the last years in the ICDAR and ICFHR conferences.. This contributed to the emergence of better and novel approaches in each edition.

Due to the vast number of works addressing DIB and any of its sub-tasks and related works (enhancement, image restoration, contrast normalization), we will go through the most significant and illustrative approaches. In any case, one could rely on surveys or other sources to get a better insight on the topic, for example, the above mentioned contests DIBCO and H-DIBCO are a good source to know the last advances in the field. A complete survey (at that time) was collected by [P. K. Sahoo et al. 1988], and later updates by [Mehmet et al. 2004] and [Sankur et al. 2001].

The deployed models in this PhD Thesis could be classified as local methods, and they are meant for images of documents, specifically historical records. We applied supervised techniques based on discriminative models. Hence, we try to emphasize on the closest related works and the differences among the followed approaches.

2.1.1 Taxonomy

We show different ways of classifying the different works to address DIB. In each classification, we introduce and analyze some of the most representative works.

Global/Local/Hybrid thresholding The general image binarization (or thresholding algorithms) could be divided into global, local thresholding and mixing strategies. Global thresholding establishes a fixed threshold for all pixels of the image, whereas local thresholding applies different thresholds for each pixel (using pixel local information). Hybrid approaches try to compose both strategies: they use global and local information for thresholding. Global thresholding techniques usually need less computation, and they work well in simple cases, though fail in complex images. The most popular global image threshold is Otsu's method [Otsu 1975] which after computing the gray level histogram, takes the optimal value that minimizes the inner class variance. There are some refinements based on this method [Brink 1992; Farrahi Moghaddam et al. 2012; Kittler et al. 1985]. The most popular local thresholding algorithms are Niblack [Niblack 1985] and Sauvola [Sauvola et al. 2000]

filters. For each pixel, Niblack takes a rectangular pixel-centered window and adapts the local threshold using the average (μ) and standard deviation (σ) of this region. The threshold value is computed as:

$$T(x,y) = \mu(x,y) + k \cdot \sigma(x,y) \ , \tag{2.1}$$

where k is the influence of the region deviation in the local threshold. As we can see, this method is not parameter free and requires to tune k and the input region size (typical values are around 25×25 squared windows and $k = 0.6$). Sauvola adapts the contribution of the standard deviation by the dynamic range R:

$$T(x,y) = \mu(x,y) \cdot \left[1 \ + \ k \cdot \left(\frac{\sigma(x,y)}{R} - 1\right)\right] \ . \tag{2.2}$$

As the authors pointed out, typical values are $k = 0.5$ and $R = 128$.

Another similar approach is the one developed by [Bernsen 1986], they compute an adaptive local threshold as the mean of the minima and maxima brightness values (intensity level) of a w rectangular window. However, if the contrast is higher than a certain threshold (the difference between higher and lower values), the pixels of the window are classified to background or foreground according to the class that fits better.

On the last modality, hybrid approaches, we address the works of [Chang et al. 2008; Farrahi Moghaddam et al. 2012; Ntirogiannis et al. 2014a; Pai et al. 2010].

Technique Grouping by technique or procedure is also a typical classification, but, most of the time it is hard to enclose a method in a single category since they pool and combine different techniques. Some of the categories we could found and representative examples are shown:

- **Clustering Analysis** [Kittler et al. 1985; Otsu 1975].
- **Histograms and image variance models** [Dos Anjos et al. 2008; Niblack 1985; Sauvola et al. 2000]
- **Entropy-based methods** [Pun 1980; P. Sahoo et al. 1997; A. Wong et al. 1989].

And other approaches like fuzzy logic [Lopes et al. 2010], and the Machine Learning (ML) that we will discuss later.

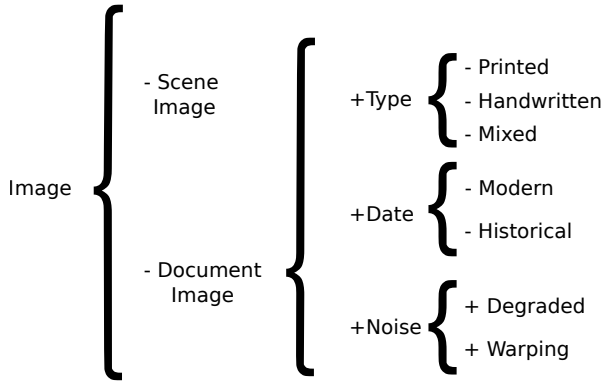


Figure 2.2: Image Binarization modes. Nodes marked with "+" are not exclusive.

Type of images On Scene Images which contain objects, usually the ratio between background/foreground is lower than document images wich present more unbalanced data and background copes the major part of the image. Due to the nature of the documents and regularity in writing, it is reasonable to have a particular research field aiming these documents. General purpose thresholding algorithms should perform worse in documents since more accurate models can exploit their characteristics. In any case, given the hierarchy, as seen in Figure 2.2, more general models, could be used in particular tasks. Indeed, most of the general purpose approaches are adaptive or parametrized, and they can conveniently be tuned for many different images. For example, Sauvola is an adaptive method that could perform reasonably well with text, degradations, and bad illumination issues. These general approaches suffice in several document tasks that do not present high deterioration.

Taking a closer look to document specific contributions, [Ntirogiannis et al. 2014a] combine global and local threshold for handwritten text recognition. First, a background estimation is applied to binarize the image globally, then they work with Connected Components (CCs), the approach takes stroke characteristics and discards components that do not suit to the common stroke criteria. The method proposed by [Farrahi Moghaddam et al. 2012] works in a similar manner since it extracts features such as stroke width and line height. It adapts itself to the document structure because it combines a grid-based modeling and the estimated background map.

A note on the data that is taken as input. The images could be coded in *RGB* or grayscale, but also MultiSpectral Images (MSI) which are obtained applying different filters of illumination (*UV*, *VIS-NIR*) generating a rectangular lattice

for an image. It is not common to find available MSI corpora [Easton et al. 2004; Shippert 2003] but this modality is gaining interest as shown in the ICDAR 2015 contest [Hedjam, Nafchi, et al. 2015]. Another type of inputs that are not taken into account in this classification images are taken from videos sequences [Roy et al. 2012; Z. Zhou et al. 2010].

Historical and Degraded Documents In degraded documents, such historical records[2], traditional and general methods present poorly performances. Degraded documents involve not only binarization but the restoration of document parts (mainly lost strokes), for this purpose usually stroke characteristics are integrated in the binarization. It is possible to find a vast set of contributions addressing these issues [Amudha et al. 2012; Antonacopoulos et al. 2007; Fung et al. 2010; Jagroop Kaur et al. 2014; Rani et al. 2015]. For example, [Farrahi Moghaddam et al. 2010] treat binarization of degraded documents by combining different binarization scales. Their work aims to restore lost strokes and eliminate ink bleed-through. [Nina et al. 2011] showed that a recursive Otsu's thresholding could be used for binarization of historical documents as well. [B. Su et al. 2013] applied another adaptive approach by the combination of image contrast, gradients, and text shapes are estimated by Canny's edge extractor. [Valizadeh et al. 2012] extract useful characteristics in order to classify the pixels in background or foreground; they use structural contrast taking into account information like the stroke width for feature extraction. With the selected features, space is partitioned into several regions (clustering) where the pixel within each region will be classified mixing the region information and the previously features extracted.

There is a full set of specific models focused on the ink bleed-through particular noise. In this concrete sub-task, we found two modes: blind and non-blind (the ink bleed-through of a page is part of the opposite recto/verso page). The non-blind approach takes as input recto/verso pairs of images while in the blind each page is given independently. The former mode seems easier and allows to deal with dirtier documents, but also arises new challenges such as page alignment. We are not going into details with this particular mode; but some related works are [Baronia et al. 2013; Drira 2006; Hanasusanto et al. 2010; Huang et al. 2008; Wolf 2010].

[2]Degraded and historical documents are two different concepts, even though they are heavily correlated since the historical documents suffer from age degradations.

Machine Learning models In this PhD Thesis we have used ANN which are discriminative models. Regarding ML approaches we mostly find works based on Markov Random Fields (MRF)/Conditional Random Fields (CRF) and other supervised methods that rely on labeled data for training the model. We found several works on CRFs for different DIA related tasks [H. Cao et al. 2009; S. Z. Li 2009; C. Wang et al. 2013]. With CRF the binarization is formulated as the problem of maximizing the a posteriori probability based on given previous observation (dirty pixels). In these models, the conditional probability of a pixel of the image depends only on the pixels of a predefined neighborhood (or clique). The model parameters (or potentials) are learned by minimizing the energy function for a fixed size clique. Even though there are several works for DIB based on CRFs they differ in the conditional probability formulation, prior estimation, and the optimization procedure.

For example, [Wolf and Doermann 2002] use MRFs for low quality text binarization/restoration, the clique potentials from the models are learnt from training data. A probability density function is defined by adding Gaussian noise with zero mean and variance (σ_n^2), then they added the Sauvola's threshold (T) to this function. The estimation is computed upon 4×4 grid clique, this big size window arises the problem of search all the possibles clique potentials (2^{16}). For that, the unseen cliques are smoothed. Finally the Bayesian maximum a posteriori is maximized by simulated annealing. In [Lelore et al. 2009] they model the conditional function as the observation of background and text distribution data. Nevertheless, they propose alternative heuristic rules. The final parameter estimation and optimization is done by an Expectation-Maximization (EM) algorithm. [Kuk et al. 2008] define the likelihood probability using a Gaussian noise estimation taking illumination and text parameters. Then they optimize the posterior probability by the graph cut algorithm. [Wolf 2010] adapted CRF models to deal with ink-bleeding images, using two different CRFs for the recto and verso sides. The primary goal is to recover the recto pixels and remove the verso pixels, with the double modeling they could identify recto pixels covered by verso ones. In [Lettner et al. 2010] the authors had apply CRFs as well but using MSI as inputs. The model combines spectral and spatial features based on the stroke properties. Then the model is trained by using Belief Propagation but also performed some experimentation with the graph-cut algorithm. Another approaches based on CRF are [H. Cao et al. 2009; Gupta et al. 2006], where patched-based topologies instead of pixels as connectivity source are used.

Connectionist approaches have also been applied for image processing due to their ability to learn very complex non-linear input/output relationships from

examples. In particular, MLPs have been used for binarization and image enhancement. [Marinai et al. 2005] collects a set of ANN-based works for DIA, they explore from document pre-processing to recognition, going through topics like layout analysis and character segmentation. ANN, specifically MLPs, have been used for the DIB task as well. For instance, in [Hidalgo et al. 2005], a MLP fed with the pixels of a fixed size moving window is presented, the net is a regression model used to enhance the image document. In [Chi et al. 2001], a MLP is trained as a classifier in two classes, also considering a sliding window running through the input image. These approaches require from supervised data to train the regression models, and hence, the training data quality has a direct impact on the overall system performance. [Mehrara et al. 2009] proposed a similar setup, but they extract edges in images. First, they apply an algorithm to obtain the possible edges, and finally, classify them in 16 different types. An extreme approach is the work of [Wu, Rawls, et al. 2015], where an ANN for pixel classification is used but using several higher order features as input. The idea is quite smart since they some features extracted from other methods, for example, the Otsu's and Sauvola's thresholds. A novel approach is to treat the problem as 2D sequences and apply RNNs to classify the elements of the sequence. We can found our collaboration in [Afzal et al. 2015], were we applied MDLSTM trained with Backpropagation-through-time (BTT) on image patches.

In some other examples on ML, for instance, [Hedjam, Moghaddam, et al. 2011] propose a maximum likelihood optimization using a high recall map by Sauvola. Besides, they restore the lost strokes combining the maximum likelihood with prior information. A Bayesian model using Hidden Markov Models (HMMs) is proposed in [F. Su et al. 2007] for removing bleed-through degradation. In [B. Su et al. 2010] pixels are classified in three classes: background, foreground and uncertain pixel; the uncertain pixel are re-classified following an unsupervised *k-means* based procedure.

Combination of different approaches We have reviewed, mentioned and, of course, excluded lots of binarization methods. Combination of several thresholding models, tandems or Recognition Output Voting Error Reduction (ROVER) are an inevitable line of research to improve unique methods. Indeed, most of the approaches evaluated try to pool as much as useful information coming from different sources. That is the case of [B. Su et al. 2011], following their previous work in [B. Su et al. 2010]. They compare the various outputs from different methods. The pixels that are not unanimous in all the methods are marked as uncertain and then reclassified by using contrast information and surrounding neighborhood. [Arruda et al. 2014] com-

pute weak and strong bi-level images by combining the structural contrast images with 2 different parametrization of Niblack method. [Badekas et al. 2007] use a self organizative maps for combining several binarization approaches: Otsu's, Bernsen's, Niblack's, Sauvola's and others. [Khankasikam 2013] states that there is not an optimal binarization technique suitable for all documents. Hence they propose a selection algorithm where an MLP is trained to chose the best binarization method in each case from from: Niblack's, Bernsen's, Kapur's, Kittler's and Sauvola's. As features, they use histograms, mean and standard deviation. The main drawback is that the selection is made at page level instead of pixels or patches.

Other approaches, like the one presented in [Rangoni et al. 2009] applies different methods with different parameters, and then the authors validate them with a set of representative pages (or even lines) and apply the best method to the full document (or collections of documents).

2.1.2 Performance and efficiency

There are also several lines of research focusing on fast and efficient computation. For example, image histograms could be efficiently optimized by using *Integral Images* [Bradley et al. 2007; Nicolaou, Ingold, et al. 2014; Shafait, Keysers, and T. Breuel 2008].

[Pai et al. 2010] propose a fast but high-performance binarization algorithm that is suitable for mobile devices. It divides the images into several blocks and uses local and global thresholding. As part of the current PhD Thesis we integrated our convolutional MLP in an Android Mobile application [Adelantado-Torres et al. 2014; Pastor-Pellicer, Castro-Bleda, et al. 2015].

2.1.3 Evaluation

Evaluation of binarization is an open problem. As we will see in Chapter 4.4 there are several approaches for evaluating the performance of the binarization stage available. The most naïve approach is the subjective evaluation, other is to compute the final OCR output or specific binarized metrics.

[Ntirogiannis et al. 2008] proposed a supervision framework by extracting the skeleton of the text. Then the user can correct the mistakes; and the final estimated ground truth is taken from a dilated skeleton of the text. [Ntirogiannis, B. B. Gatos, et al. 2013] later proposed a set of evaluation techniques aimed

for historical documents, they considered precision and recall but also the influence of missed text, background and stroke reconstruction.

2.2 Text Line Extraction

[Nagy 2000] completed a great overview about DIA. Even though this survey is more than 15 years old, it captures the essence of the DIA tasks and the progress over the twentieth-century. In this PhD Thesis we have focused as well in one of the most exciting and crucial stages which is Text Line Extraction (TLE). This stage is part of the overall pipeline, and usually, it goes preceded by other steps like layout analysis, page skew correction, and image denoising. In this section we collect related works to the models presented in this PhD Thesis (Chapter 5) and other illustrative approaches to get a better understanding about TLE and how to approach this problem.

First of all, one has to address the brilliant survey by [Likforman-Sulem et al. 2007], they grouped and reviewed the leading works on Historical Documents. The different approaches have been classified as follows:

- **Projection profiles** As the name states, these models sum up vertically the contribution of all the pixels and by analyzing this histograms (finding peaks for example) the lines are detected and extracted. As a chief drawback, they usually fail with short lines and narrow space between them.
- **Smearing** The Run-Length Smearing Algorithm (RLSA) procedure by [K. Y. Wong et al. 1982] is applied horizontally and then the lines are segmented. Similar ideas have been used in Chapter 5.6.
- **Grouping** This technique includes most of the bottom-up approaches like the methods proposed in Chapter 5.
- **Hough transformation** Translating the lines, points and other components to the Hough domain, [Hart et al. 1973] allow to find and check potential line alignments and therefore to extract them.
- **Repulsive-Attract** This category, as far as we know, only includes the work from [Öztop et al. 1999]. The idea is to set up several baselines along the text lines and find their exact location by attracting the baseline and foreground pixels and repelling them from other baselines.

- **Stochastic approaches** Most of the ML approaches are probabilistic models since the classifiers learned and estimated probabilities given the input images.

Other classification are proposed. For instance, [Ouwayed et al. 2010] presented a new categorization with three top levels: (1) Top-down: projection-based and document-based methods; (2) Bottom-up: K-NN, Hough transform, Smoothing, Repulsive-attractive Network and, Minimal Spanning Tree methods and; (3) Hybrid approaches.

In [Razak et al. 2008] after introducing some of the common challenges of TLE (line fluctuation (skew) and line proximity), the authors reviewed several approaches grouping them by smearing methods, projection based, grouping, Hough and graph based. [Bukhari et al. 2009] proposed a much higher lever classification dividing it into two types: connected components and deformable model based. We also recommend checking the review included in [Fernández-Mota et al. 2014].

2.2.1 Bottom-up strategies and text aggregation

To approach the problem of noise and styles variation in handwriting, the majority of published methods rely on bottom-up strategies grouping low-level elements of the image such as pixels or components. Simple rules such as nearest neighbor fail since it often belongs to an adjacent line. Typically, low-level parts such as text segments are computed as CCs [Feldbach et al. 2001; Louloudis et al. 2008] or contours of the image [Romero, Pastor, et al. 2006].

In [Yin and Cheng-Lin 2007; Yin and Liu 2009] a bottom-up graph based procedure by joining CCs with minimal spanning tree clustering is followed. The key point of this approach is to define a distance measure between components and find clusterings in the minimal spanning tree to segment each line. [Ouwayed et al. 2010] instead, aims at multi-oriented documents. In addition to multi-skewed lines, we find multi-oriented lines within the same document. The procedure has been studied on Arabic documents, but they claim that it can be generalized to another script where writing is linear. Their approach starts with a document meshing to allow several orientations, then active contours (snakes) take the morphological information and CCs of the same line are grouped for each mesh. The orientation estimation of each cell is calculated by means of projection profiles and the Wigner-Ville distribution. The final lines are extracted by enlarging the meshes to their neighbors with similar orientation until obtaining the lines. A smart touching components splitting

is applied by using morphological information detecting the touching points between ascenders, descenders, and Arabic ligatures.

Interest Point (IP)-based methods circumvent the problem of binarization and offer the advantage of being independent of the actual layout, IP extracted from geometrical methods, such as Difference-of-Gaussian, have been previously used to detect text lines in historical documents [Garz, Fischer, Bunke, et al. 2013]. First the points are grouped into text segments using spatial clustering and finally joined in to lines.

2.2.2 Projection profiles

With respect to projection profiles, [Bar-Yosef et al. 2009] applied them to degraded documents with large skew and curvatures for TLE. The local profiles are computed at column level using a sliding window. Finally, a local projected profile for every pixel is obtained and the seams are calculated by first-order derivatives. [Dos Santos et al. 2009] use morphological operators to extract features that will be combined with local projection profiles. This approach comprises 8 flow stages: 1. Feature extraction by the morphological operation; 2. Vertical Histogram Projection, 3. Text Line Separation, 4. False Line exclusion, 5. Line region recovery, 6. Histogram projection, 7. False Word exclusion, and 8. Text Selection.

Projection based models assume uniform skew within the lines. If it is not the case, piece-wise projections are applied instead. These ideas are very close to the contribution presented in Chapter 5.5.1 since our methods allow different skews within lines and different orientations for each text segment. [Papavassiliou, Stafylakis, et al. 2010] for instance, divide the page into non-overlapping equal-width vertical zones and after disregarding the zones with a proportion of foreground pixels below a certain threshold (margins), a smoothed projection profile for each vertical zone is applied. Text and gaps are distinguished from the profiles and refined by a HMM which two states: text and gap regions. Finally, separators between lines are estimated. The key point relies on joining the separators of the contiguous columns and a post-processing refinement to avoid line overlapping.

2.2.3 Blurring and smearing

Lots of contributions apply smearing in their pipelines which are lately combined with CCs or projection profiles among others. [Nicolaou and B. Gatos 2009] use a blurring procedure to deal with touching components, line markers are extracted in the limits between lines. Next CCs are assigned to one of these lines. If a CC belongs to many lines (touching components), the pixels are rearranged according to the line areas extracted from the markers. [Nikolaou et al. 2010] applies a modified RLSA by adding CCs, white spaces, punctuation marks, and skeleton of the strokes knowledge to the smearing process. In [Bukhari et al. 2008, 2009] the authors tackle the text segmentation problem by using active contours (snakes) as a base unit to minimize the energy function. The final lines are extracted by joining neighboring snakes after applying several deformations until they stick together. To retrieve the "snakes", first a multi-oriented anisotropic Gaussian filter bank is used for the image smoothing, and then ridges are taken as the central lines from the smearing.

[Shi, Setlur, et al. 2009] combine blurring and morphological information. The smearing procedure uses a local connectivity map, which smooths the data within an elongated ellipse as pixel mask. "Each pixel value in an ALCM image represents the cumulative intensity of the foreground pixels in an elliptical neighborhood around the pixel in the original document image. A pixel with higher value in the ALCM implies that the pixel is in a dense text region." Then the map is binarized and joined in CCs. Finally, the touching components are split using the method proposed in [Shi and Govindaraju 1997]. [Papavassiliou, Katsouros, et al. 2010] apply morphological operations such as erosion, dilation and m-th rank order operations to extract the structure of the lines (the image is dilated and smoothed after applying 1×8 and 8×1 erosion operators before downsampling the resultant image). A median order filter removes the ascenders and descenders and keeps the MBA of the text. The resultant CC are post-processed using more structuring elements to eliminate touching components. Another inspiring work is the one depicted in [Y. Li et al. 2008]. In there, first they introduce some of the TLE evaluation metrics discussed in Chapter D.1. The approach computes a probability map marking whether a pixel belongs to a line. The Probability Density Function is calculated by means of Anisotropic Kernels. Once we have the "blurred" probability map, the problem is reduced to find boundaries between lines. The probability map looks like creeks and valleys (similarly to the MBA maps generated in Chapter 5.6) and the final segmentation is performed by the level set method. Put it

Figure 2.3: Text Reference lines.

in short, “the initial boundaries evolve by its partial derivative and an external vector field”.

2.2.4 Line seams

Another trend is to detect the separation (seams or boundaries) between lines. We found some methods following the idea of the Seam Carving algorithm [Avidan et al. 2007] which is widely used in computer vision and it was initially meant for image resizing, but it can be easily adapted for image segmentation. It compute seams of minimum energy through an image, and this is calculated efficiently by DP. The base idea behind applying it to TLE is to determine the energy function that will found spaces between lines. [Saabni et al. 2011] use the signed distance transform to generate the energy map. [Arvanitopoulos et al. 2014] apply a modification of the seam carving procedure with text line constraints. “Medial seams” are computed by projection profile matching and the seam carving method is constrained to each of this medial seams. Seam carving could be utilized as well, for post-processing stages, like in [Garz, Fischer, Sablatnig, et al. 2012] where it is used for splitting touching components when a line overlap has been detected.

[Gao et al. 2011] draw paths between text lines in a multi-scale approach. After several pre-processing the average character size is computed (they address Chinese writing), and then they combine 3 different sources of information: simple local minima algorithm by checking the foreground pixel distribution on regions above and below each pixel, CC contours and piece-wise projection profile.

2.2.5 Text Reference Lines and baseline detection

Another set of approaches explores the text line characteristics such the reference lines which are explained in more detail in Chapter 5.2 (illustrated in Figure 2.3). In addition some works that compute the text reference lines for text line normalization are discussed in Section 2.3.

Tracking the text reference lines is a good approach to guess the location, continuity and orientation of the text lines, so they could be used in bottom-up approaches to aggregate text segments into lines. For example, [Feldbach et al. 2001] compute the mean and baselines of text using extrema points; then they extract lines detecting the main baselines of the text.

An alternative idea is to track the MBA, in this modality, [Baechler, Liwicki, et al. 2013] classify the pixels of the image to belong to the text core (MBA) line class. They perform two classification levels; first, pixels are classified as decoration, background, text block or periphery classes. And then, with a higher resolution input, the outcome of this first stage is used for a more fine classification into decoration, background, and core-text line. The detection of the MBA at pixel level have been used for Text Line Normalization (TLN) already used for TLN [Pastor-Pellicer, España-Boquera, Castro-Bleda, et al. 2015] and presented in Chapter 6.

2.2.6 Touching components

Touching components between lines is one of the principal problems of TLE where most of the approaches are not robust enough when it is strongly present. Therefore, there exists a full line of research addressing this issue. Indeed, most of the works reviewed in here include their particular solution to the touching components issue included in their solutions. Touching components poses two main problems:

- Detection of the touching lines that must be splitting: this is usually straightforward if the line spacing is wide or consistent. For example, compute the average line height and mark to split the lines that are above certain threshold. The problem becomes harder when the space between lines is narrow, or the lines present different height or are skewed.
- Splitting the lines: most of the touching cases come from the overlapping of descenders and ascenders of two consecutive lines. The cut separating the lines must pass through ink strokes.

[J. Kumar, Kang, et al. 2011] introduce an approach for segmenting handwritten Arabic text-lines in the presence of touching components. This work is an improvement of the method previously described in [J. Kumar, Abd-Almageed, et al. 2010]. It comprises four main steps: coarse text-line estimation, error detection and correction, touching component localization and separation. To

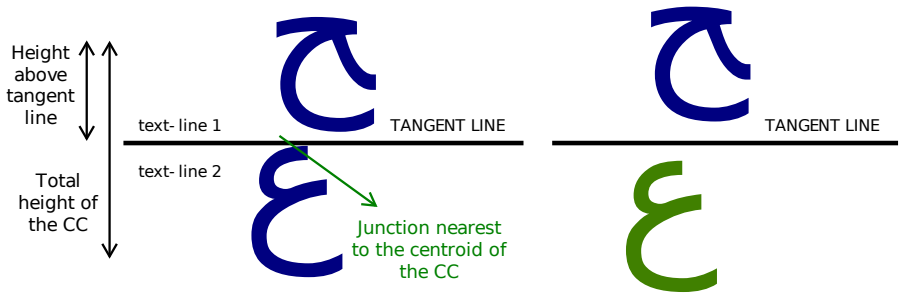


Figure 2.4: This example illustrates how a text component is split by comparing the height of the component above the tangent line with the height of the whole component. The image of the right shows the outcome after separating the junction nearest to the centroid of the CC (Image taken from the report done by Georg Schaller for the DIA Seminar at University of Fribourg). The report was focused on work developed in [J. Kumar, Abd-Almageed, et al. 2010].

localize the touching components they took the upper and lower tangent lines of two consecutive lines: "If the ratio of the length of component below or above this tangent line to the total height of the component is greater than some threshold then the component is considered to be a touching component.", as illustrated in Figure 2.4.

[Kennard et al. 2006] address the problem of touching components in historical documents by applying several methods. After pre-processing, they use the foreground/background transitions to estimate text lines. During the process it provides the foreground data for each line as well. Hence, it removes noise for the further text line processing stages. A different approach to overcome splitting components to the detect the potential touching points. Usually, character overlappings appear between ascenders and descenders of consecutive lines. [Kang and Doermann 2011] collected different touching template patches in a dictionary; then the potential touching components are identified, and the splitting is resolved as the correct segmentation in the retrieved closest patch from the dictionary.

[Rohini et al. 2012] aim at segmentation of touching and skewed documents. Touching components are detected by extracting core text regions through horizontal profiles. Then the line separators are drawn, and touching com-

ponents are fixed by using RLSA. For the segmentation errors derived from skew, they use distance measures.

2.2.7 Text Line as sequences

Related to the main line of this PhD Thesis, we found the work of [Moysset et al. 2015]. In their approach MDLSTM are trained with Connectionist Temporal Classification (CTC) for the localization of line frontiers. The line extraction is done at paragraph level and requires from skew correction first. For training, only the number of lines is provided, so the CTC algorithm will try to align a sequence of line/interline transitions for each row. This approach has their counter part by using HMM for text line segmentation in [Bosch et al. 2012]. They adapted the work of [Z. Lu et al. 2000] to HWR. As Moysset's work, the algorithm is applied to single unskewed columns. The image is aligned vertically to 4 possible regions: Normal text Line-region, Inter Line-region, Blank Line-region, Non-text Line-region. The model finds the best region sequence $\hat{h}$ as

$$\hat{h} = \arg\max_{h} \sum_{b} P(o, b|h) \cdot P(h) \ .$$

where b is the alignment and o the observations at each row. The first term is estimated by an HMM[3] by using the Viterbi search algorithm. The syntactic model $P(h)$ defines the a priori class of the given sequence; this term constrains the sequence to have a valid structure. For the observation probabilities, a mixture of Gaussian is trained. The image is converted into sequence of feature vectors by dividing the image into D non-overlapping vertical regions, then projection based features are extracted for each block and finally stacked for each row. In this work, they used a HMM to estimate the sequence alignment and the input features of the model are handcrafted by projection profiles, while in [Moysset et al. 2015] used the CTC and MDLSTM that are directly applied on the raw image.

[3]There is a more detailed explanation of HMM in Appendix B.

2.2.8 Historical Documents

Historical documents present more degradation due to the conservation state, and sometimes their structure or layout are more complex than other modern texts. Historical documents are gaining more interest in the literature in the last years due to several reasons:

- By definition, they remained undigitized and nowadays most of the libraries are scanning and transcribing their databases. Newer and more powerful techniques are aiming this daunting task and every day it is easier to get more accurate transcriptions, which increased the interest on this field of research.
- Historical documents are more challenging which promotes the creation of novel and more powerful techniques. The interest of the research community into overcoming most of the challenges and issues is rising.

Many works could be found in this field. [Diem et al. 2013] aim different type of documents with several layouts, orientations, and backgrounds. They follow a bottom-up approach where the CCs are grouped into words and then into lines by globally minimizing all words distances. Skew and binarization pre-processing are required and then the words are joined by using Local Projection Profiles. To speed up computation, a word is enclosed into a simple rectangle area. Then the orientation of each word box is computed by extracting the upper and lower contours, two respectively straight lines are fitted by regression. If both lines had similar direction the box is oriented to them, if not the box follows the general skew orientation of the document. Then the words are merged according to their minimal distance; the distance takes into account: the Euclidean distance of upper, lower and middle points between boxes, their angle difference and a constant C that depends on of the text. Some inconsistent words are removed from the text line orientation.

[Fernandez et al. 2012] introduced a line segmentation method based on pathfinding (A*) from background skeleton. The algorithm is compared with [Manmatha et al. 2005], they use the same word segmentation but different line detection method. Later, works like [Fernández-Mota et al. 2014] detected seams between lines by finding paths in a graph. The possible lines are located first and then segmented. The segmentation is reduced to a graph traversal problem by finding potential starting and ending nodes and minimum cost path between them. “The paths consist of background points at equal distance to the word above and below”. The nodes of the graph cover the background area and touching text lines are treated by adding virtual edges in the graph.

As they claim, the technique does not require any learning process, and it is not oriented to a specific writer, style or alphabet.

[Cohen et al. 2014] use anisotropic filters for detecting the lines. They approximate the probability density function of the lines (high probabilities in the center of the lines and little in the frontiers) by a Gaussian filter. A multi-scale filters are set up to make the method scale-invariant. In a second step, they binarize the outputs of the Gaussian filters and extract CCs by using a component tree. For each element obtained, a kind of center skeleton/polyline from left to right on the component is drawn. From those, a fitness function states whether this element is a text line. As always, some final post-processing eliminating small components and other spurious artifacts are applied. This approach gave excellent results on historical documents, especially in the Saint Gall and Parzival Datasets which we have used for evaluation of our proposed models as well.

[Rabaev et al. 2013] assume that working on binarized images fails for most of the TLE algorithms when the documents have active degradation. Working on the gray scale images, they propose a two staged approach: first, they extract potential characters candidates in the image by using evolution maps of CCs [Biller et al. 2012], in a second step the potential characters are aggregated into lines using a sweep-line approach. Sweep lines move following the writing direction and when an element is detected it is assigned to one of the lines or discarded. These contributions gave good results in historical documents, moreover some examples of very degraded documents where it is possible to recover the lines are shown.

2.2.9 Machine Learning

Most of the works discussed so far are parameter free or adaptive, while others require continuous parameter tuning. There is another subset that can learn this parameter from the given data. For example, the contributions shown in Chapter 5 use generally ANNs in order to estimate text zones (or areas).

However, most of these approaches combine ML techniques with heuristic and pre and post-processings. For example [Baechler, Liwicki, et al. 2013], described previously, uses Dynamic MLPs to tag part of the images. The sequence segmentation approaches showed earlier [Moysset et al. 2015] and [Z. Lu et al. 2000], learn the sequence model parameters from the input data and the number of lines in the paragraph. [Stafylakis et al. 2008] followed

a similar approach for the interline modeling with sequential models. First, the image is split into several fixed-size columns to overcome the problem of (non-uniform) skew. Then they use the typical scenario: horizontal projection profile and smooth plus minima and maxima values for segmentation. With this, over-segmented candidates are taken and finally refined with a Viterbi model with two states: gap and text line.

[Kang, J. Kumar, et al. 2012] codebook based text line segmentation procedure is followed. The image is summarized (downsampled) in smaller $p \times p$ bins. At bin level, the codebook is formed by a contextual k neighborhood centered in a bin and the mask indicating which contextual bins belong to the same line to the center bin. The codebook samples are clustered, and during evaluating, each bin is classified to one of the clusterings which are used for a first coarse text line segmentation which is post-processing to get a final text line segmentation.

2.3 Text Line Preprocessing

Once text lines are extracted, a new set of processing stages are applied directly to those lines. Text line pre-processing comprises several stages (Figure 2.1). The order of these stages is important since some processes are assumed in following steps. For example, a naïve text size normalization step assumes that the slope and skew have been corrected. Otherwise, unwanted artifacts could appear. Besides, some transcription engines require height line normalization as input while others do not. Even though, some techniques could skip some of the pre-processing steps where input data does not suffer from high distortions or postpone its treatment to later stages.

It is common in the literature to find works pooling all the preprocessing steps at once instead of individual studies for any of them. Indeed, most of end-to-end HWR engines present their own approach based in previous normalization works [Bozinovic et al. 1989; Brown et al. 1983; Bunke et al. 1995; Buse et al. 1997; Caesar et al. 1995; Guerfali et al. 1993; Plamondon et al. 2000; Vinciarelli et al. 2001]. However, some of the more challenging stages which are skew and slant, have particular works dedicated to them. Besides, it is worth to mention that the work developed aims at Western scripts, or at least scripts that share some of their properties such as the presence of ascenders and descenders.

2.3.1 Tracking Text Reference Lines

Most pre-processing modules comprise the detection of the text zones or areas as explained in detail in Chapter 5.2. In the literature we could find different ways to track text reference lines:

- **Histogram Profile based** [Burr 1982; Guerfali et al. 1993; Hennig et al. 2002; Powalka et al. 1994; Vinciarelli et al. 2001].
- **Run-Length Smearing Algorithm (RLSA)** [Pastor-Pellicer, Afzal, et al. 2016; Toselli, Juan, et al. 2004].
- **By contours/extrema** [Bozinovic et al. 1989; Pastor et al. 2004].
- **By Local Extrema** [Brown et al. 1983; Caesar et al. 1995; Gorbe-Moya et al. 2008; Guerfali et al. 1993].

The vertical histogram projection profile is the most straightforward approach since the middle zone uses to concentrate more ink than the ascenders and descenders. Geometrical heuristics may fail in many cases, specifically in short sentences or isolated words. Since they are based on ink statistics, they may be confused in the presence of too much or too few ascenders and/or descenders, and they can also be affected by the presence of long horizontal strokes. Local extrema-based approaches extract the vertical maxima and minima of the strokes; these points, usually, belong to one of the 4 reference lines. One basic approach is to use the horizontal projection histogram for this purpose [Brown et al. 1983; Guerfali et al. 1993], or ML techniques relying on supervised or semisupervised data [Gorbe-Moya et al. 2008]. The reference lines can be parametrized as straight vertical lines, diagonal straight lines [Caesar et al. 1995], parabolic [Caesar et al. 1993] and also intervals [Gorbe-Moya et al. 2008; Guerfali et al. 1993; Powalka et al. 1994]. [Caesar et al. 1995] estimate the baseline by linear interpolation of Local Extrema Point (LEP), selecting the subset of baseline points using regression analysis.

2.3.2 Skew and Slant correction

Skew correction is usually applied to the whole page instead of individual lines or words [Y. Cao et al. 2003; Hull 1998]. The problem arises when there is non-uniform skew within lines. In this case, the skew correction needs from text line segmentation and line splitting into words or text units and individual treatment of each one. At line level, [Sun et al. 1997] apply histograms for detecting the skew direction and the slant. They use a CC-based approach by identifying and transform the surrounding parallelogram. [Morita et al. 1999]

minimizes the weight least squares on the convex hull on the mathematical line morphology. [Marti et al. 2002b] analyze HMM recognition systems with statistical language modeling, they stack several preprocessing steps following the ideas of [Brown et al. 1983]. For skew correction, they detect the baseline but assuming that the lower baseline could be approximated with one straight line. For slant, vertical contours are extracted and approximated by vertical lines. The angle histogram of this vertical lines is collected, and the final slant angle is the maximum value of this set. [Vinciarelli et al. 2001] proposes a skew and de-slant approach without relying on heuristics. Horizontal projection histograms are used for the text core region detection, w.r.t slant correction they compute the angle that maximizes the number of columns with continues vertical strokes. [Buse et al. 1997] explore two different slope/skew corrections. One applies several angles and takes the projection histogram, but they compute the minimum entropy of the whole distribution. The second approach translates the image to the frequency domain. [Kavallieratou et al. 1999] explored several ways of detecting the skewed angle: by Cohen's class distributions on the horizontal profile and then combined with the Wignner-Will distribution function as in [Kavallieratou et al. 2002]. In fact, some of the techniques used in Chapter 5 were inspired by this work.

There also exist contributions mainly focusing on the slant removal procedure. This is the case of [Pastor et al. 2004] where they compute the angle with maximum variance on the vertical projection profile for each de-slanted image. This work is similar to [Slavík et al. 2001] in which Sobel filters are computed to calculate the best slant angle. [Bertolami, Uchida, et al. 2007] uses a local slant correction instead of averaging the slant of each word/line. An alternative is to use ML to estimate the MBA avoiding geometrical heuristics. [Seiler et al. 1996] uses an ANN to obtain a rough estimate of the text-core zone. It is important to remark our previous work [Gorbe-Moya et al. 2008] where the slant is applied non-uniformly by using DP. For this, the slant is corrected independently for each column but restricting the change over columns to a certain degree. They use also supervised methods with ANNs to estimate the score of each angle in each column.

2.3.3 Size Normalization

Some recognition engines require fixed column height since the features are commonly extracted at the column level. Size normalization often implies width normalization, nevertheless we focus mainly in height normalization.

On the one hand, the advantages of having a fixed height are twofold: reduces the script variability and allow to extract a constant number of features for each image column (or context of columns). On the other hand, some of the features extraction models and recognition engines do not require height normalization and sometimes bad scaling could distort the images and therefore decrease the recognition rate. When dealing with variable image heights, for extracting a fixed set of features one could apply a non-uniform sliding window [Bianne-Bernard et al. 2011; Kozielski, Forster, et al. 2012], or pooling features in upper levels [Graves and Schmidhuber 2009].

Models that rely on tracking reference lines use different scaling for each zones preserving the size of each for the whole set of lines. [Toselli, Juan, et al. 2004] gives a non-linear height normalization, given 30% of the final size to ascenders and 15% according to the average ascenders and descenders distribution of the corpora. Similar to this ideas, in [Gorbe-Moya et al. 2008] (and in Chapter 6.2), 10% and 20%, have been taken for ascenders and descenders respectively.

An interesting approach is the normalization based on second order moments [Casey 1970; Miyoshi et al. 2009]. [Kozielski, Forster, et al. 2012] applies moment-based normalization to HWR corpora such as IAM and RIMes database. They run a variable window length in the image to scale by using moments. In this procedure, the image is normalized but also centered on its gravity center. Indeed, moments are used, as well, for feature extraction.

Width normalization It is slightly different than height normalization since the variance between lines are mainly due to the sentence length and not from writer variability. Therefore instead of normalizing the whole sentences width to a fixed size, most of the approaches try to standardize the width variance per character or stroke. Is it possible to perform width normalization by counting, for instance, the average number of changes of white-space/ink, in horizontal, in the MBA rows and using this value, calculate the average character size in the training data. [Graves and Schmidhuber 2009] tries to approximate the mean character width to a fixed value, they take the text middle line (the line between baseline and corpus line) and normalize the width by the stroke crosses in this line. [Marti et al. 2002b] counts the black-white

transitions in the horizontal direction. Thus the average number of pixels per stroke is estimated. Finally each image is re-sized to match the average pixel per character. For isolated words or scripts like Arabic, we could normalize each word to a fixed width keeping their aspect ratio [Dreuw et al. 2009].

On-line Text Line Normalization It is interesting to remark some works on on-line text normalization. One can always render the stroke sequence into a 2D image and apply some of the mentioned normalization. But, for instance, it is straightforward to compute LEP from the strokes. In [Graves, Liwicki, et al. 2009], the baseline and corpus line are calculated by linear regression of those extrema in two steps where in the first one out-layers are discarded. It is interesting also to address some of the complete on-line recognition engines for on-line HWR like [Jaeger et al. 2001] or the Google multi-language HWR [Keysers et al. 2016] where they also present the smart pre-processing in this modality.

2.4 Handwriting Text Recognition

In this document, when talking about HWR we generally refer to unconstrained off-line handwriting recognition, which is one of the most challenging modalities, since there are not temporal relations between the strokes, and it has to deal with large vocabularies. Word and character segmentation in this mode are difficult and most approaches rely on the use of large dictionaries and Language Models (LMs), even though the number Out-of-Vocabulary (OOV) word occurrences is still very significant.

One of the goals of this PhD Thesis is to improve the performance of our previous HWR engine[4]. At line level, we have also invested some effort in pre-processing, in particular, text size normalization, using detection of text line zones. The following natural step to improve the overall handwritten recognizer is to update the optical models, based on Hidden Markov Models hybridized with ANNs (HMM/ANN), by using novel deep learning techniques.

In this section, we collected some useful related works on HWR, focusing on optical modeling and feature extraction.

[4]The original HWR engine developed by the group is presented in Appendix B and it has been used in the experiments for Chapter 6.

Other important parts of the overall transcription process are language modeling and decoding algorithms. We skipped these from the reviews since they are out of the scope of this PhD Thesis.

2.4.1 Optical Modeling

One naive approach for HWR is to split the sentence into words, or even characters, and then classify each word/character independently. However, there is a cycle dependency between segmentation and classification tasks known as the Sayre's paradox [Sayre 1973]. To overcome this problem, HMMs does not rely on an explicit segmentation. They are generative models where the observed output might be seen as the contribution of many possible segmentations although, in practice, only the path of best probability is computed by means of the Viterbi algorithm. HMMs have been successfully applied to several sequence labeling problems [L. R. Rabiner 1989].

HMMs comprise transition and emission probabilities $P(x|q)$, the most used emission model are Gaussian mixture models (GMMs). An alternative is to use discriminative models that compute the $P(q|x)$ and can be turned on emission probabilities following the Bayes theorem.

$$P(x|q) = \frac{P(q|x)P(x)}{P(q)}. \tag{2.3}$$

Although, when using this value for obtaining the word sequence which best explains the observed input we can ignore the term $P(x)$ since these terms are the same for every case of the maximization and they do not depend on the word sequence which is the value we are maximizing. Put in other terms, we can use scaled values:

$$P(x|q) \propto \ P^{\star}(x|q) = \frac{P(q|x)}{P(q)}. \tag{2.4}$$

Hybrid systems are one of the most successful approaches to HWR where ANNs compute the emission probabilities of the HMMs, namely: MLPs in [Espana-Boquera et al. 2011; Senior et al. 1998], CNNs in [Bluche, Ney, and Kermorvant 2013b], RNNs in [Marukatat et al. 2001], and combinations of them. Additionally, we can find other related models: Radial Basis Functions in [Singer et al. 1992], Support Vector Machines (SVMs) in [Stadermann et al. 2004], or

time-delay networks in [Caillault et al. 2005; Jaeger et al. 2001; Schenkel et al. 1995].

An alternative sequence modeling to HMMs is the well-known Connectionist Temporal Classification (CTC), which was introduced in [Graves, Fernández, Gomez, et al. 2006]. It allows sequence labeling to be performed: CTC aggregates the contribution of every alignment and adds a new *blank* grapheme/-phoneme output to the net. This technique is mainly meant for RNNs and has generated many successful works, particularly in HWR and speech recognition, among others.

Nevertheless, RNNs can be combined with both HMMs [Schenk et al. 2006; Senior et al. 1998] and CTC. It is worth noting the work of [Graves, Liwicki, et al. 2009] where Hidden Markov Models with Recurrent Neural Networkss (HMM/RNNs) and CTC are compared. LSTM [Hochreiter and Schmidhuber 1997] have shown to successfully tackle the vanishing gradient problem derived from long time recurrences. In addition, Bidirectional Long Short Term Memories (BDLSTM) [Baldi, Brunak, et al. 1999; Graves and Schmidhuber 2005; Schuster et al. 1997] allow the sequence to be scanned by two RNNs, one from left-to-right and another from right-to-left.

Deep learning techniques [G. Hinton, Deng, et al. 2012] make easier to work directly on the raw signal (2D images in the case of HWR). Other approaches use GMMs combined with ANNs in tandem (the posterior probabilities are taken by the GMMs). For example, in [Grézl et al. 2007], an MLP bottleneck is used to extract features for the tandem. This approach was later improved by Deep Belief Networks (DBN) as seen in [Sainath, Kingsbury, et al. 2012].

As Figure 2.5 shows, the different techniques can be combined, and basically all of them have been explored for HWR and speech tasks. Nevertheless, there are some restrictions and tendencies. For example, CTC is most efficient when used with RNNs [Bluche, Ney, Louradour, et al. 2015] and is actually mainly applied to LSTM networks.

When using HMM/ANNs, the MLP classifier usually receives a contextual set of handcrafted features for each frame of the sequence [Espana-Boquera et al. 2011; Marukatat et al. 2001; Senior et al. 1998; Toselli, Juan, et al. 2004].

Some other recognition systems directly receive the raw image as input. For example, CNNs and MDLSTM are convenient approaches for raw input signals since they deal with 2D sequences properly, as in [Bluche, Ney, and Kermorvant 2013a; Graves and Schmidhuber 2009]. Hierarchical sets of MLPs

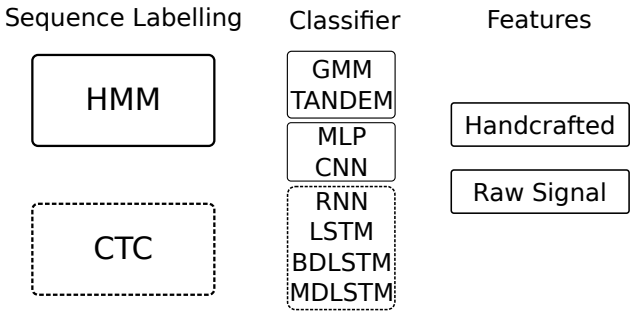


Figure 2.5: Connectionist sequence labelling models. There are two main tendencies: HMMs and CTC. The connectionist models estimate frame posteriors to get likelihoods in case of HMMs and the labelling sequences in the CTC. GMMs estimate the emission probabilities in the original no connectionist approach.

are used to extract features from raw pixels in [Dreuw, Doetsch, et al. 2011]. They apply nested MLPs in the HMM/ANN system, and then they combine them in tandem with GMMs. However, more recent works have used deep architectures: [Bluche, Ney, and Kermorvant 2014] use deep MLPs for optical modeling, with up to 9 hidden layers. A complete comparison between handcrafted features and pixel inputs is also performed. In addition, deep models are compared with BDLSTM, and dropout is applied in both architectures. The best results are finally obtained by means of a ROVER combination of the four models.

The methodology followed in our experiments was somewhat explored in [Bluche, Ney, and Kermorvant 2013b], whose CNNs estimate the state posteriors of the HMM. In that work, two different approaches were followed: the first one performs grapheme classification after segmentation, whereas the second one uses a sliding window to perform the sequence alignment. A tandem approach with CNNs and GMMs was also explored in [Bluche, Ney, and Kermorvant 2013b].

In [Kozielski, Doetsch, et al. 2013], they propose a discriminative training of HMMs in tandem with BDLSTM and GMMs. Principal Component Analysis is also applied to the extracted features, leading to 20 features per frame. They got very competitive results due to the maturity and performance of all the transcription components: a smart feature extraction relying on previous moment-based image normalization [Kozielski, Forster, et al. 2012]; variable number of states per character obtained from initial alignment statis-

tics; writer adaptation applied on GMMs; and a combined word and character mixed language model to overcome the OOV issue.

Finally, for the feature extraction process, most of the features are based on applying a sliding window to extract values such as gray densities, background/foreground estimation, and their derivatives [Bianne-Bernard et al. 2011; El-Hajj et al. 2005; Kozielski, Forster, et al. 2012; Marti et al. 2002b; Toselli, Juan, et al. 2004].

2.4.2 Feature extraction

Finally, regarding the feature extraction process, most of the features are based on applying a sliding window comprising cells in order to extract some values such as gray densities, background/foreground estimation and their derivatives [Bianne-Bernard et al. 2011; El-Hajj et al. 2005; Kozielski, Forster, et al. 2012; Marti et al. 2002b; Toselli, Juan, et al. 2004].

2.4.3 Trends for HWR

As a general observation, most of the techniques applied in HWR have been previously developed for Large Vocabulary Continuous Speech Recognition (LVCSR) and adapted to HWR afterwards. Thus, current speech approaches can be expected to be transferred to HWR in the near future.

The use of deep models in LVCSR is well established. Most of them still require handcrafted features, which is the case of [Seide et al. 2011] and [Veselý et al. 2013], where different features combined with hybrid HMMs and DBNs for weight initialization are studied. [G. Hinton, Deng, et al. 2012] gives an overview of deep ANNs with many layers to produce HMM posterior probabilities, using pretraining techniques that are based on Restricted Boltzmann Machines, which are also presented in [A. R. Mohamed et al. 2012]. This novel architecture uses up to eight hidden layers trained as DBN.

However, there is an increasing trend towards the use of raw signals instead of handcrafted features. [Dahl, Yu, et al. 2012] use context-dependent deep models that work directly on the spectral speech signals. They later improved their results by applying Rectified Linear Unit (ReLU) and dropout [Dahl, Sainath, et al. 2013]. Other end-to-end approaches include the works of [Hannun et al. 2014] and [Amodei et al. 2015], where RNN models trained with CTC are used.

The incorporation of CNNs in hybrid HMMs for speech has been quite recent as seen in [Abdel-Hamid et al. 2012], where CNNs have been successfully combined with HMMs. [Sainath, A. R. Mohamed, et al. 2013] rely on acoustic parametrization, studying the number of convolution and fully connected layers keeping the number of learned features constant for each model. [Saon et al. 2015] go further by combining CNN with recurrent models that are conveniently trained with dropout and maxout units.

More recent works explore deeper convolutional architectures [Bi et al. 2015; Sercu et al. 2015], which are known as Very Deep Convolutional Networks [Simonyan et al. 2014]. More than 14 hidden layers are stacked, convolutional kernels are usually 3×3 shaped, and pool layers are interleaved with 2 or 3 convolution layers.

Another interesting line of research is the sequence-to-sequence models [Sutskever et al. 2014], which has been successfully applied to translate tasks [Cho, Merrienboer, Bahdanau, et al. 2014; Cho, Merrienboer, Gulcehre, et al. 2014] and are successfully combined with attention-based models [Dzmitry Bahdana et al. 2014; Wu, Schuster, et al. 2016]. Indeed, the tendency is moving towards to avoid recurrent connections making them more (or at least partly) parallelizable [Gehring et al. 2017; Vaswani et al. 2017]. These models have also been used in speech [Chorowski et al. 2015; L. Lu et al. 2015] since the nature of speech (and handwritten) is a sequence-to-sequence transformation.

2.5 Summary

In this chapter the state-of-the-art for the main DIA tasks dealt in this PhD Thesis are presented. We discussed several of these approaches and focused on the most relevant and illustrative techniques. We analyze also the closest and the most inspiring works to the ones presented in the current document.

Chapter 3

Neural Networks

Contents

One of the goals of this PhD Thesis is to explore the use of connectionist models for Document Image Analysis (DIA). We have used different ANNs models such as MLPs, CNNs or even RNNs for several DIA tasks and in almost all cases the ANN receives an image as input.

In this chapter, we do not go deep through the ANN basic formulations and other mathematical proofs. We assume that the reader has some basic knowledge about ANNs. Nevertheless, we address some illustrative works to get a better insight into the field. Since ANNs are a transversal topic in this PhD Thesis, we will collect some of the common ideas and techniques used during training and inference of our ANN models. We will show the procedures, especially parameters and hyper-parameters optimization and regularization techniques, that are shared by the several nets trained in the different stages presented in the following chapters.

Finally, we encourage the reader to check periodically, in the literature and many online tutorials, the significant advances in deep learning that, for sure, will help to improve the techniques discussed in here.

3.1 ANN basics

As its name depicts, ANNs try to imitate the way of the human brain learns and solves problems. Nevertheless, the traditional implementation of ANN differs significantly from real neurons, notwithstanding they still share the concept of neurons and connections between them. An ANN is made up of neurons (process units) and weighted connections between them. It can be seen as a mathematical model where each unit takes new values depending on its connected neighbors and the internal activation functions (Figure 3.1). In the traditional approach, the neural network has an entry point (input) and then, the neurons compute their values according to these inputs. Finally, an output neuron (or set of neurons) generates the "answer" given the input. The rest of neurons are referred as "hidden".

The real power of ANNs is that they can learn complex functions from data. The weights (or parameters) of the net are learned during the *training* phase and are used to compute the outputs of the model during *inference*. The most basic set of ANNs is known as *Forward Neural Networks* which do not present cycles. The most well-known case is the Multilayer Perceptron (MLP) [Rosenblatt 1958]. In an MLP, neurons are organized by stacked layers, the values of one layer are fed to the next layer (Figure 3.2).

Forward pass We will assume that we already have a trained ANN, and we want to use it for the problem we are dealing with. For example, classification of handwritten digits from 0 to 9, (which is the MNIST task). Our input in this scenario would consist on a 16×16 image input (concatenated in a layer of 256 input values). The output consists of 10 neurons, where the neuron o_i estimates the probability of output class i (or c_i), given the current input. Hence, our model classifies the sample to the digit that gives us the major probability ($c_i = \arg\max_i o_i$). For each neuron, during the forward pass, its value is computed as the weighted sum of previous layer activation values (with the addition corresponding to the bias, see Figure 3.1).

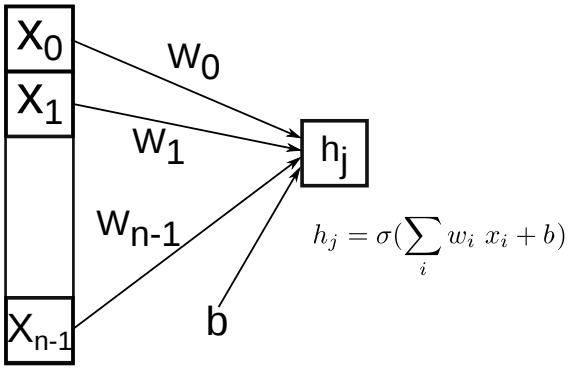


Figure 3.1: Hidden neuron computation. The input value depends on the weighted sum of its inputs values, then an activation σ function is applied.

Activation function An activation function is applied to each neuron after the weighted sum of the previous layer. Most of the activation functions can be found in many external sources (`https://en.wikipedia.org/wiki/Activation_function`). The activation functions mainly used in our experiments are:

logistic It has been used by default for the hidden layers in most of the experiments carried out. The logistic function is also applied to the output neurons when we have a classification task of 2 classes ($\frac{1}{1+e^{-x}}$).

tanh The hyperbolic tangent has a similar to the logistic one, but its output values are in the range $(-1, 1)$ ($\frac{2}{1+e^{-2x}} - 1$).

ReLU The Rectified Linear Unit (ReLU) is a simple but effective activation that helps to avoid the gradient vanishing problem as we will see in Section 3.4.

softmax This function is mainly used in a multi-class classification task, since it normalizes all values of the layers in the range $(0, 1)$ according to each neuron contribution [Costa 1996]. This could be seen as a probability estimation since all the values sum-up 1 ($\frac{e^{x_i}}{\sum_j e^{-x_j}}$).

3.2 Training the networks

The error Backpropagation (BP) algorithm is the most used approach to train ANNs. It is combined with a higher optimization algorithm (e.g. gradient descent). In short, this procedure starts by taking the desired output (ground truth) and the predicted value, and it iteratively computes the gradients of the weights by the chain rule. For that, we need all the operations applied in the net (from inputs to outputs, weight operations and activations) to be differentiable. *Loss* is the function that computes the error between the ground truth and the predicted output and it should be derivable as well.

Weight initialization The weights are updated by the BP procedure, but an initial weight configuration is required for the first first. Indeed, and unfortunately, the weight initialization has a direct impact on the net performance since it could lead the optimization algorithm to an undesirable local minimum in the function to optimize. Usually, the weights are initialized to random values sampled from a uniform or Gaussian distribution. Those values should be relatively small in order to have small gradient steps and avoid problems like exploding gradients (Section 3.4). This initialization could be conveniently tuned with *fan-in* and *fan-out* normalization, which consists on normalizing the weights according to the number of input and output connections in each neuron. Usually the initial values are normalized by the square root of the fan-in and fan-out as follows:

$$w_i = N(\mu, \sigma)/\sqrt{n_{in}, n_{out}} \ . \tag{3.1}$$

This initialization ensures that all neurons in the network have approximately the same output distribution and empirically improves the rate of convergence. More extensive information about weight initialization could be found in [Glorot et al. 2010]. Finally, it is worth to mention that layerwise pretraining, as we will discuss about in Section 3.4. It can be applied to have a more suitable initial weights when the final supervised training is performed. But in any case, even for the pre-training stage, the weights has to be initialized at the very beginning by one of the methods discussed here.

Stochastic Gradient Descent This technique for training ANNs is also well-known, and it has been successfully applied in several pattern recognition tasks (classification, regression, forecasting, ...). Different weight updating modes exists [Duda et al. 2001]:

- the *off-line* training mode (also know as full batch), which computes and sums the derivatives of all training patterns and updates weights once every epoch;
- the *on-line* training mode, which calculates the derivative of one training pattern and updates weights once for each pattern every epoch; and
- finally, the *mini-batch* or *batch* training mode, which computes and sums the derivatives of a few training patterns, updating weights once for the mini-batch size, but several times for one epoch. [1]

Mini-batch and on-line training modes have some advantages compared with the offline mode: convergence is faster and the result is equal or even more accurate. This mode is known as Stochastic Gradient Descent (or incremental gradient descent), and it has been used to train all the networks showed in this report. The data (or samples) are split into random sets (a.k.a mini batches or bunches [Bilmes et al. 1997]) and incrementaly apply the weight updates incrementally for each.

The size of the bunch has not only a direct impact in training but also in the system's efficiency. Since most operations (weight matrices products) can be performed with algebraic operations, most of the toolkits take advantage of that and perform bunch computations in parallel (especially with Graphic Processor Units (GPUs) computation). Sometimes bigger bunches make the training/inference faster, but sometimes, we have to restrict the size of the batch to get better results during trainer. Note that during inference, since no weights are updated, the output values do not depend on the bunch size. During our experiments, the batch size is one of the hyper-parameters that we have to tune during our experiments, but usually, we try to keep it around 32.

[1] In some bibliograpy and other resources, they refer to the size of the mini batch also as "bunch". We have used mini-batches during all the experiments of this thesis, in some cases we refer to the size of the minibatch as "batch size" or "bunch size" indistinguishly.

Replacement The data set is divided into random bunches, and the weight updating is performed for each of this mini-batches. Usually, the training set is divided into two sub-sets: train and development (or validation) and usually at the end of each epoch the error is computed for the development set. During training, once an *epoch* is completed over all mini-batches the net is evaluated with the development set. We talk about replacement when, instead of using all the train data to complete an epoch, only a subset is used befor the evaluation step. If the training data is vast and redundant selecting only a portion of it for each iteration will speed up our training/validation procedure. Summarizing, having a set X_T with T training samples and a batch of size B, a replacement is a random subset X_R of size R, that is $X_R \subset X_T$. Therefore, $\frac{R}{B}$ weight updates are applied in each epoch. Replacement is often applied also to the development set. But in this case, we have to be careful, since the validation set is not invariant anymore. The stopping criteria rely on the loss/error of the validation set, and this score is not strictly comparable since the samples are different. If this subset is large enough, the error estimation is similar to the entire validation error, and it could still work for validation purpose. For instance, we used random replacement on the validation set on our HMM/ANN model for HWR, since the number of frames is enormous.

An alternative, used in some of the *one-to-one pixel labelings* tasks, is to use a fixed step on the validation set. Since pixels neighbors are usually similar, an advance step (or stride) will keep the validation set almost invariant, and at the same time, we reduce the number of forwards operations by a factor of the step size.

3.2.1 Stopping criteria

The iterative training algorithm stops when no further improvement is expected. One could setup a fixed number of epochs (even bunch iterations) and keep the weights of the net that has given best performance on the development set.

A standard approach is to “continue training while the error in the development set is still decreasing”. So the termination criteria are based on the development set error since the training error (or loss) could be still decreasing due to over-fitting. Usually, an absolute number of epochs without improving is used prior the training termination. The problem comes if we got a false good result in early epochs and training finishes much sooner than it should. This issue is easily solvable by defining a minimum number of training epochs, where the validation losses are ignored. Another approach is to use a relative

number of epochs without improving; the net would need fewer epochs to develop in early stages but more in later stages. Usually, the training stops in the epoch (best_epoch$(1 + \alpha)$), where α is the relative improvement parameter. The problem here is that the training could take too long since we need to wait for more epochs for each improvement. But in any case, this could be combined with an absolute criteria (training will stop once one criterion is accomplished) or we can always restrict the maximum number of epochs.

Regarding to the improvement criteria, an absolute definition is to consider that the model is improving if the development error is strictly lower than the best so far. But with adaptive optimizers the improvement gets smaller and smaller and it is not worth it to keep training, so it is convenient to add an absolute or relative threshold.

In our experimentation we have usually used one of the criteria described here, there is not a universal answer, and at as most of the cases it depends on the task. We cannot always train as long as we want because of computational restrictions, so the faster and earlier the net converges, the better.

Continuous evaluation In other cases, the training and evaluation processes are detached, and they are performed in paralel. The evaluator process reads the weights of the model in a determined steps and performs the operation in a different thread. It is not required to wait for a training epoch to finished to evaluate the dataset, and the training is not stopped while the evaluation. Indeed, several trainers could work in the paralel and then a master process will join the gradients of each servers [Dean et al. 2012].

3.3 Architectures

One can find many different ANN architectures and variants in the literature. In this section, we introduce some of the models used in this work, which mainly involves: Multilayer Perceptron (MLP), Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

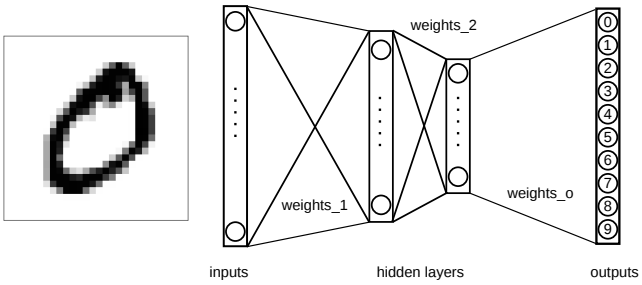


Figure 3.2: A Fully Connected MLP. Input consists of 256 input neurons. Hidden layers receive previous values. The output layer performs the classification of the input sample in one of the $0 - 9$ classes.

Fully Connected Multilayer Perceptron These topologies correspond to the initial forward models previously mentioned. The neurons of each layer are "fully" connected to the neurons of the previous layer. Figure 3.2 illustrates an MLP for the digit classification task (MNIST).

Convolutional Neural Networks CNNs [Lecun et al. 1998] are a kind of deep neural networks biologically inspired by the visual brain cortex and how the cells are arranged in layers. The cells in one layer are sensitive to a small region of the input, called receptive field, and these areas are tiled to cover the whole input space. Different layers are connected sequentially to extract useful information from the input.

In mathematical terms, this process is described as a convolution between the whole input space and a kernel matrix. Convolutions with different kernels are computed together to produce several output maps, which are different transformations of the input. This convolution allows the extraction of local features which are invariant to translation in the input space. The dimensionality of the input space constraints the dimensionality of the convolution, being a 1D convolution in case of time-series data, 2D convolution in case of grayscale images, 3D convolution for color image or grayscale videos, and 4D

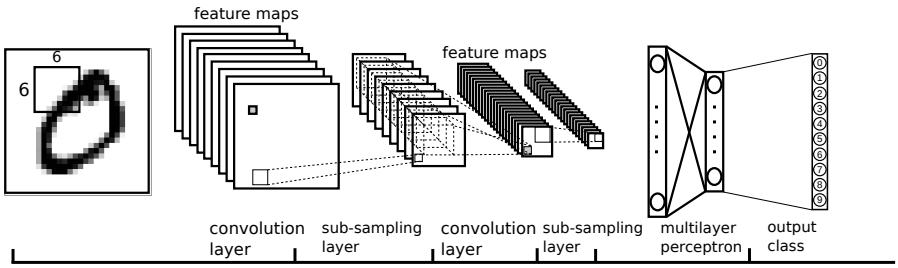


Figure 3.3: Basic CNN configuration for the MNIST task. The input is one map of 16×16 pixels, and the following convolutions are applied in the 2D space. After several convolution layers, the features are flattened and fed to an MLP. Following fully connected layers are applied after the convolutional feature extraction.

convolution for color videos. Convolutions of subsequent layers may traverse the set of maps previously computed.

To reduce the dimensionality of hidden layers, pooling operations are applied to the output maps after a convolution layer. Different pooling strategies have been proposed in the literature [Krizhevsky et al. 2012; Sermanet et al. 2012; Zeiler and Fergus 2013]. Regarding the advantages of more complex pooling layers, max-pooling is widely used in CNN systems. Since we are focusing in DIA task we have mainly worked with 2D convolutions. The third dimension defines the number of maps (or channels).

Generally, a CNN sequentially combines several layers of convolution/activation/pooling, acting as a deep extractor for high-level features. The output of this convolutional part is fed into a standard MLP, being the output of this MLP linear for regression or classification tasks. Figure 3.3 shows a basic configuration for MNIST with CNNs.

Recurrent Neural Networks RNNs are meant to classify sequences. Convolutions are somehow dealing with sequences since the same neural network is applied to different steps of a sequence. Nevertheless, they do not keep any state information from previous steps. Recurrent connections are added to the model, so in the next step, the previously hidden state is fed to the new input into the current (Figure 3.4).

RNNs can be seen as deep nets in time, and therefore it is hard to keep long term dependencies as stated in [Bengio, Simard, et al. 1994; Hochreiter 1998]. Long Short Term Memoriess (LSTMs) have been the most success-

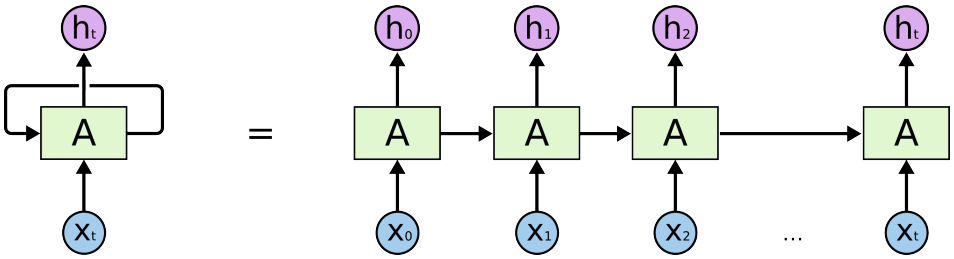


Figure 3.4: The figure (right) represents the time unrolled network. The figure has been taken from `http://colah.github.io/posts/2015-08-Understanding-LSTMs/` and we encourage recommend to read the article for a better understanding about RNNs and LSTMs.

.

fully neural "cells" used in recurrent nets to deal with the vanishing gradient problem [Hochreiter and Schmidhuber 1997]. Bidirectional Recurrent Neural Network (BDRNN) are an extension of RNNs where a sequence is scanned in both directions: forward and backward and both passes are combined to generate each step output [Schuster et al. 1997]. It is possible also to stack several BDRNN to produce a deep BDRNNs [Graves, A.-r. Mohamed, et al. 2013], and of course, it is possible to combine them with convolution layers.

Indeed, RNNs are also extended to n-dimensional sequences: MultiDirectional Long Short Term Memories (MDLSTM) [Graves, Fernández, and Schmidhuber 2007]. With respect to our tasks the images are 2D sequences, so a RNN runs through the image. Particularly, the pixel $x_{i,j}$ of the image, the recurrence takes states $h_{i,j-1}$ and $h_{i-1,j}$. Actually, there are 4 different directions to scan the image: top-to-bottom and right-to-left, top-to-bottom and left-to-right, bottom-to-top and right-to-left, bottom-to-top and left-to-right. Likewise BDRNNs the 4 layers are combined to generate the outputs. Generalizing, in a n-dimensional sequence it is needed 2^n RNNs in order to scan the image in all directions.

For training RNNs, the BP has to be slightly tweaked to work with time recurrent relations. The algorithm known as Backpropagation-through-time is the most widely used [Werbos 1990]. Sometimes the sequences are too large, and it is memory and time inefficient to backpropagate such long dependencies. In these cases, a common practice is to truncate the BTT to a fixed number of time steps [Williams et al. 1990]. Also, this allows for training with fixed length bunches and improving the parallelism during training.

3.4 Regularization

In this section we collect some of the regularization techniques aplied while training neuronal networks which allowed to overcome the most common neural networks training problems:

- Over-fitting: when the model learns the particularities of the training data and performs poorly for unseen data. In these cases, the training error keeps improving while in the development/test set the error gets worse. The opposite term is “generalization”.
- Vanishing gradient: when updating the weights by the chain rule, the “front” layers are slower to train.
- Convergence: we look for models that are fast and robust to train, which involves two main factors:
  - Training speed and computational efficiency.
  - Reaching a good state (results) in fewer iterations/epochs.

Data normalization It is convenient, and almost mandatory, to normalize the data. Normalization makes the training and inference of the network more efficient. Straightforward normalization methods are Min-Max, which reduces the feature range to $[0, 1]$, or Gaussian normalization, to $[-1, 1]$.

Most of the input data used in this PhD Thesis consist of raw image pixels, where the values are integers in the range $[0, 255]$. It is easy to apply Min-Max normalization by dividing by 255. Indeed, since 0 denotes black pixels, we invert the values $(1 - x)$, so darker pixels have high activation values. In our approaches we normalized the data as follows:

$$x_{norm} = 1.0 - \frac{\text{float}(x)}{255} \quad . \tag{3.2}$$

Some other works follow a Gaussian normalization which implies computing the mean and variance from the training data.

Add distortions and noise to the input of the image One could generate new artificial data or add noise on the net input to improve generalization and reduce unseen data. In some tasks, adding Gaussian noise or filters like Salt-Pepper are good enough to improve the overall training. But with raw document images, Gaussian noise does not cover the differences between characters or even writers. For this purpose, elastic distortions, as well as scale and rotating deformations of the image inputs, will provide more realistic general data.

Weight decay One of the essential regularization techniques is to add the sum of the norm of the weights into the loss function. In this case, high weights are penalized and the models try to keep them low:

$$loss' = loss + \alpha \sum_i (w_i) \ . \ \text{(L1-norm)}$$
$$loss' = loss + \alpha \sum_i (w_i^2) \ . \ \text{(L2-norm)} \tag{3.3}$$

$L2$-norm adds the squared contribution of the weights, and it is known as weight decay, (α is known as *weight decay penalty*). It is usually set to very low values and it has been used almost in all the experiments in this PhD Thesis.

Weight Max norm With weight decay, we are minimizing the overall weight contribution in the loss, but it is still possible to have some high weights. One option is to constrain the maximum absolute weight values. In that case, the weights are truncated. Usually, we used values between 4 and 8.

Rectified Linear Unit (ReLU) The ReLU activation function helps to avoid the vanishing gradient problem. This a great advantage front *sigmoid* or *tanh* functions. In some of our models convergence is hard to achieve. ReLU activation allowed to train faster and had better results in our tasks.

Dropout Originally proposed in [G. Hinton 2014; G. E. Hinton, Srivastava, et al. 2012], dropout involves randomly removing some hidden units of an ANN during training but keeping all of them during testing. More formally, consider a layer with d units and let h be a d-dimensional vector of their activations. When dropout with *droprate* probability p is applied at this layer, some activations in h are dropped. During inference, all units are retained, but their activations are weighted by a factor of p. Dropout involves a hyper-parameter

p, for which a common value is $p = 0.5$. It could be not only applied to fully connected layers but also convolutions, when usually a smaller p values are needed [Deng et al. 2013; Krizhevsky et al. 2012; Seltzer et al. 2013].

Dropout helps to avoid overfitting, especially when the model has lots of parameters, and it has been successfully applied to several tasks like Speech Recognition [Dahl, Sainath, et al. 2013; J. Li et al. 2013] or HWR [Pham et al. 2014]. There are some extensions like DropConnect [Wan et al. 2013] where the weights are dropped instead of the neurons activations. Other useful sources to understand Dropout are [Baldi and Sadowski 2013; Iosifidis et al. 2015; Srivastava 2013; S. I. Wang et al. 2013].

Layerwise pretraining A trend for weight initialization is to precompute the initial values using an unsupervised approach. The main idea underneath is to try to extract the more discriminative features from the inputs before the discriminative training. There are two main approaches lines are Restricted Boltzmann Machine (RBM) [G. E. Hinton and Salakhutdinov 2006] and Stacked Denoising Autoencoders (SDAE) [P. Vincent et al. 2010].

In this PhD Thesis, if pre-training has been applied, we followed the second approach (SDAE) by adding Gaussian and Salt-Pepper noises in the layer activations. The weights of each layer are trained like an unsupervised autoencoder, where the input of each layer is the previous layer input with some distortions, and the output is the non-degraded input. Once the weights of a layer are trained, the next layer is trained with the outputs of the previous layer.

Clip gradient Another problem when training, especially in RNNs, is the *gradient explosion* [Bengio, Simard, et al. 1994]. In each training step (minibatch update) many gradients are accumulated. Long term derivatives tend to have higher derivatives, leading to a significant increase of the norm of the gradients which makes the training very unstable.

The most applied solution to solve this cumbersome is to clip the gradients if they exceed a *threshold*. Hard clipping where the derivates are kept below the threshold or soft where the new gradients are computed like:

$$\delta w = \text{threshold} \frac{\delta w}{|\delta w|} \ . \tag{3.4}$$

Batch and layer normalization Another powerful idea that could improve convergence is to normalize the inputs of each layer, especially for deep architectures such as Batch Normalization [Ioffe et al. 2015] and Layer Normalization [Ba et al. 2016]. In the first approach, the inputs of each layer are normalized w.r.t each batch; during inference the parameters are normalized to the overall training dataset stats. The next approach follows a similar idea, but data is normalized according to all the neurons in the layer, the advantage here is that the same normalization is applied between train and inference.

Other trends Every year one could find novel and better regularization techniques which also speed up convergence. Some techniques worth to mention are, for example, *Curriculum Training* [Bengio, Louradour, et al. 2009], *Maxout* networks [I. J. Goodfellow et al. 2013], where the new units take as output the maximum value of a set of neurons or residual nets [He et al. 2015].

Recently Generative Adversarial Networks are gaining lot of interest, citing their main work [I. Goodfellow et al. 2014]:

> [...] estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. [...] This framework corresponds to a minimax two-player game.

3.5 Deep learning

It is difficult to agree what *deep learning* is, and if what we are doing could be considered as deep learning or just MLPs with many layers. What it is true is that deep learning works, or at least in the recent years several tasks are having better and better scores since deep models are applied.

The success of deep learning comes from mainly three points that all combined started a new age in neural computing:

- New optimization techniques. Regularization terms allowed to overcome the vanishing gradient problem and train successfully deeper and deeper models.

- Each layer is seen as a new level of features representation. Having deeper models allow to have higher feature hierarchies so we can use lower features or directly raw inputs. Hence, feature extractors are avoided in some cases, since the net can learn more useful features from raw inputs. Illustrative examples are image and speech. Raw pixel values are more often used as inputs in image related task, high-frequency raw signals (spectrograms) are used as inputs of sequential models for speech.

- The use of GPUs as processing units allowed to have cheap and very powerful training resources. Indeed they efficiently allowed to perform linear algebra operations of level 2 and 3. In combination with mini-batch, GPU speeds up training and inference and made the use of the deeper models practical.

- Lots of data. Despite having models that are able to learn complex functions, tons of data are needed to train these models correctly. Nowadays, with the massive use of Internet, mobile devices, social networks; a lot of data is tracked and (anonymously?) collected in new databases. Which somehow makes that all the stated before works.

3.6 Hyper-parameters tuning

Deeper and newer ANNs and smarter optimization algorithms impose a new set of hyper-parameters [Bergstra, Bardenet, et al. 2011] that must be tuned when finding the final configuration. ANN optimizers such as Gradient Descent, Adagrad, Adam, Adadelta, RMSProp, Quickprop and for sure new ones every year, are used to train the neural weights[2]. But most of them require to set up other hyper-parameters, even the adaptive parameters which still need some seeds or initial values.

On the one hand, the main set of hyper-parameters is related to the net topology, especially deeper models that have many layers and a huge range of possibilities, as with CNNs. On the other hand, other hyper-parameters such as learning rate or momentum and regularization terms. (weight decay penalty, drop rate or even random seeds).

The straightforward way to handle it is by manual setting. Somehow it works relatively well if we have a limitation of computational resources. Since it is

[2] https://en.wikipedia.org/wiki/Stochastic_gradient_descent#Extensions_and_variants

not possible to run a broad range of configurations, a human can easily get some insights from previous runs and decide the new parameters to try. This is a commont technique when we setting up new models to get the first insight of their performance.

Grid search, which consists on the combination of all the possible values (usually discretize by intervals) for the explored parameters, works reasonably well when the number of parameters and their ranges are relatively small. The combination of the several value ranges could be easily paralleled since there are no dependencies between runs. The combination of the parameters explodes as soon as more hyper-parameters are added, making an exhaustive search a barely practicable technique.

There is a simple but effective parameter tuning which is the *random search for hyper-parameter optimization* [Bergstra and Bengio 2012]. Here the different values of the hyper-parameters are sampled according to random distributions. During this process, several configuration instances are generated, and the nets are trained with the sampled values. Finally, the setup that gets the better result on the validation set is taken. This approach is successfully combined with early stopping to avoid unnecessary computation if early losses are bad. It is very easy to implement and parallelize.

Another approach to deal with the huge set of hyper-parameters is to use Genetic and Evolutive algorithms to find promising parameter configurations [Bäck et al. 1993]. More sophisticated way of optimizing is to use statistical models such as Bayesian optimization or Gaussian processes [Snelson et al. 2006; Snoek et al. 2012].

3.7 One-to-one labeling

One of the main contributions of this PhD Thesis is the use of ANN that worked directly on the 2D raw images and provide convenient features for the different cases of study. Some of the approaches consist on one-to-one[3] pixel labeling, which has been applied to:

- **Document Image Binarization (DIB)** where each pixel is classified as background/foreground (Chapter 4).

[3]Why not many-to-many? This approach could be named as many-to-many, since the input is formed by several pixel and so do the output. However, each pixel is classified in one class, so there is a one-to-one correspondence between input and output

- **Text Line Extraction (TLE)** for the Main Body Area (MBA) detection (Chapter 5). Pixels of the MBA are marked as part of the core text zone or not.
- **Layout** where eahc pixel is classified as: text, graphics, decorations, and background (Chapter 5.8).
- **Text Line Normalization (TLN)** is performed following the same setup than the previous MBA estimation but in this case it is applied at line level for image height normalization (Chapter 6).

For our purposes we have explored three possibilities of ANNs: Fully Connected MLPs, CNNs and MDLSTM. In this section, our goal is to clarify how the different approaches work, their main drawbacks and perks, and the differences among other approaches.

3.7.1 Sliding window CNN and MLP

Each pixel has to be classified in one of several classes depending on the task: for binarization as foreground/background, for MBA in two classes or many in Layout detection. What we do is to treat each pixel as an independent sample. Indeed, the net receives contextual information around the pixel to classify. It is usually a centered window (it is possible to aggregate other features as well). Hence, an ANN runs through the image as an sliding window. Figure 3.5 illustrates the procedure. The MLPs and CNN receive the same input but they do different computation. Note that this input is the one provided by the sliding window. This is seen as a convolution since the same net is applied in several parts of the image. The main particularity of this approach is that each sample is classified independently, while CNN combines the outputs of each convolution in higher layers and then compute features of the overall image.

For simplicity, we use padding when the part of window falls out of the image. There are different pad filling that could be applied: use zeros (or white pixels), use the value of the closest pixel, mirroring the image, or even estimate the background value of the noised image (i.e. with a median filter). In our experience, we have not found big differences among approaches and the latest alternatives sometimes could present not desirable artifacts.

CNNs in their traditional set up are meant to classify one image in one of the given classes. Therefore the images are normalized to a fixed size and several stacked convolution/pools are used to extract overall features and then

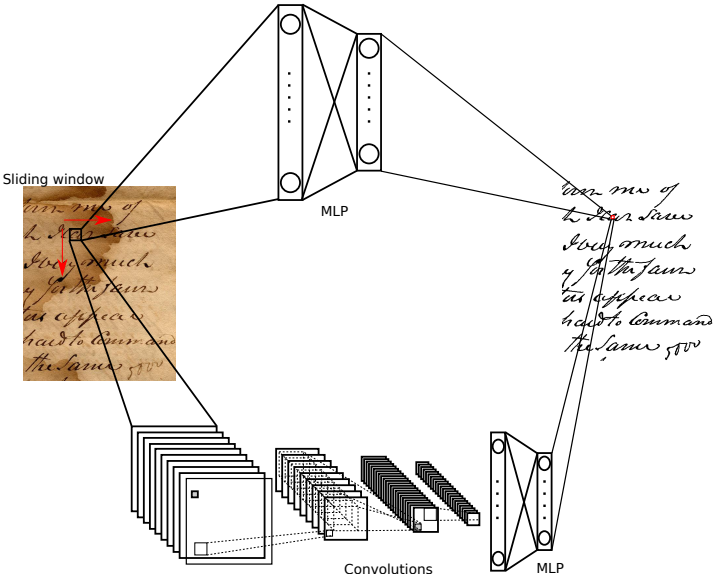


Figure 3.5: The MLPs and CNN receive the same input but they do different computation. The input in each step is provided by the sliding window.

a classifier to label the whole image in one of the classes is finally applied. Another option to avoid image size normalization is to aggregate elements in the layers before classification.

For one-to-one pixel labeling, one straightforward approach is to apply several convolutions without padding neither strides since they reduce the size of the maps. Thus, in each convolution the same *height* and *width* are kept for every convolutional map (the number of planes may variate in each convolution). Then in the last convolution all the maps are flatten to one map which correspond with the output of the net. Note, that the output has the same shape than the input, so it is possible to backpropagate the loss by agreggating all the pixels in the output. In other words, the input of the ANN is the full imatge or patch and the output is an image of the same size. However, this approach is not taking advantage of poolings (with stride), remember that poolings are avoided to keep the size of the image. If we apply several strides (in convolutions or poolings), the planes are reduced by a factor of s. Hence the planes in the last convolutions will have size $\frac{height}{s} \times \frac{width}{s}$. One common approach to deal with this is to re-size the ground truth images by a factor of s and use them as the output of the CNN. Finally, the resulting image (or class map) is resized to the original size using interpolation. Obviously, this is a rough estimation,

since a lot of information is missed during the image transformation, and it is not desirable for high-resolution tasks. Fully Convolutional Neural Networks go one step further and convert back the reduced maps into the original image using *deconvolution* procedures [Long et al. 2014]. This approach is very powerful since a new set of deconvolution kernels is used to generate back the size of the original output.

Notwithstanding this elegant solution, we have opted for the first approach (the sliding window) since in the other cases the spatial information it is reduced to smaller maps and then when restoring the original sized output some detailed information is lost. Indeed, the sliding window CNN applies a simple CNN to each window and classify it into one of the classes. This approach is much slower than others since a full net is used for each pixel of the image. It also restricts the complexity of our nets, where traditional CNNs approaches are more efficient. Therefore one of the contributions of this PhD Thesis is to make this architecture to work in a reasonable amount of time and achieve better performance than other methods.

Another perk of this approach as always other CNN-based approaches is that it is easily parallelizable. It is possible to chop the image in several independent windows (one per pixel) and process them independently in batches. Each window (or pattern) is processed independently wich can lead to unnecessary computation, yet they could reuse the maps generated by the first convolution. I.e. to compute the maps generated by the kernels of the first convolutions and use them as the input of the sliding window.

3.7.2 MultiDirectional Long Short Term Memories

Even though MultiDirectional Long Short Term Memoriess (MDLSTMs) work slightly different than the other non recurrent nets, the use of RNN is an elegant solution to perform the one-to-one labeling problem since images can be taken as 2D pixel sequences. RNNs keep the state of the previous steps in the sequence and generate an output for each input pixel.

The image sequences are not unidirectional in their nature. Indeed, it is multidimensional pixel could get contextual information from any of the 4 directions. MDLSTM add n (one per dimension) recurrence connections. These recurrence connections allow to keep several contexts in the neurons internal state, and with small input windows (0 to 3 pixel neighbors) good results could be achieved. Nonrecurrence models (MLP/CNN), in a sliding window sce-

nario, have to increase the size of the input windows significantly to extract enough contextual information in each step.

There are several drawbacks of this approach; one is the difficulty to train MDLSTM on long sequences. We work mainly with medium/high resolution images, where they have millions of pixels. The BTT algorithm unfolds the time sequences and kept the errors of each step to update this weights. For full sequences, this procedure remains impracticable since we have to bear in memory the gradients of each weight in each point of the sequence. For example, for a given pixel, tracking a little context of 32×32 pixels requires to keep the context a $32 \times 32 = 1024$ recurrence steps. To address this issue, one could limit the number of recurrence steps updated on each step or split the images in patches. Even with LSTM cells that have shown good performance in longer sequence, but with 2D recurrence sequences the internal iterations that we have to keep for having a meaningful context grown $O(n^2)$.

3.8 Summary

This chapter introduces several ANN concepts since this is one of the common topics to this whole document. Some basic ideas about the architectures used have been set up: regularization and other issues are needed to train and use ANNs that have been applied in the several tasks.

Chapter 4

Image Cleaning and Enhancement

Contents

Image cleaning and enhancement, if not the first, is one of the first steps of the Document Image Analysis and Optical Character Recognition pipeline. With a scanned document image as input, this stage discriminates between

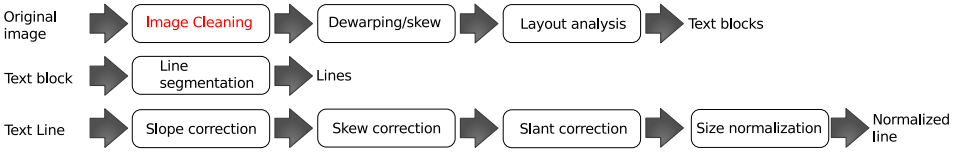


Figure 4.1: Pre-processing steps (Document Image Binarization).

foreground (which is mainly text and figures) and page background. *Text/non-text* classification is a related subtask where the text is the primary foreground information that is used in following stages.

There are subtle differences between *cleaning and enhancement*; and *Document Image Binarization*. Nevertheless, both modes have the same aim: to improve the performance of the DIA pipeline. In fact, the cleaning methods could be adapted to binarization by thresholding the output values. We will continue this discussion in section 4.2.

On the one hand, ultimate OCR/HWR and DIA systems rely on *RGB* or grayscale images, without prior binarization neither denoising, as inputs. Those methods are robust enough for a certain level of noise. Besides, in other cases, heuristics and traditional binarization methods like Sauvola [Sauvola et al. 2000], Otsu's [Otsu 1975] or background removing techniques, such as median filter, suffice. On the other hand, the interest in DIB has been raising in the last years, proof of it are the Document Image Binarization Contest (DIBCO) and Handwritten Document Image Binarization Contest (H-DIBCO) which have been hosted in the ICDAR and ICFHR conferences, respectively.

The noise presence will clearly affect posteriors stages [Likforman-Sulem et al. 2007]. Therefore, to improve the overall recognition workflow, it is interesting to use robust and adaptive DIA modules and decoding algorithms combined with an excellent image pre-processing which involves a smart cleaning and enhancement stage. An ideal system tries to minimize the noise level in the original image but also the following stages are adapted to work with previous mistakes.

In this Chapter we will analyze and explore different connectionist techniques for the DIB task. The basic idea is to receive a scanned image and output the cleaned (and enhanced) version of it. The present methods will run through the image labeling each pixel. We have started using a basic MLP based system, and then improving it by adding more features, and regularization terms during training. Following, we introduce the bases from state-of-the-art archi-

tectures like CNNs and LSTMs. And finally, we took all the approaches to find a working combination to improve previous results. The MLP-based system is a particular case of a convolution since it runs through the image as a sliding window. In this set up, each neuron of the first layer would correspond to one kernel, and following layers are seen as 1×1 convolutions. In addition, several document image corpus have been analyzed to find when is more suitable the use neural networks and other machine learning based methods, since it strongly depends on the task we are dealing and the resources available for supervising.

4.1 Supervised Methods

For the DIB task, heuristics and other adaptive approaches work well with different types of images: pictures and documents with various fonts and styles. Even though they are general, some specific tuning could improve the results for a particular domain. These free parameters usually are tuned for a small set of images and then applied to the rest of the collection. Conversely, pattern recognition and machine learning based methods require some regularity between the data seen during training and the real data.

The methods analyzed in this chapter determine their parameters directly from the seen data, and they tend to fail if there is a mismatch between the training samples and the expected data. Thus, the more representative data, the better. This premise is crucial for the task we are dealing with; a prior task analysis would help to whether is better to use ML than other techniques. For instance, if we want to clean and enhance a small set of images when we do not have a reliable ground truth and no previous knowledge about the noise, it is not worth it to use a supervised methods. But, if we expect thousand of scanned document pages which present different noises, but still show some consistency on the script; with a tiny fraction of the corpus, we could train a successful model that gives better results than other techniques for the rest of the collection.

Supervised methods have several drawbacks. As “supervised” suggests, they require enough meaningful supervised data to learn successful features from the training images. Another issue is that traditional connectionist approaches for DIB are not invariant to scales, translations or rotations. For this purpose, it is possible to apply some image rotations and distortions to extend the training data with artificial samples. One could, for instance, perform a particular set of noisy samples, and create new data with more meaningful patterns. The

model is trained with the original data, and then the samples that have higher error are taken to generate new training data. Bootstraping techniques as we will see help to overcome these problems.

Pixel-to-pixel labeling Generating clean images from dirty ones involves to classify each pixel in one of the main classes: foreground, background. In our methods we treat each pixel as a sample to label, hence for a small set of images, we have thousands and millions of individual samples. The ANNs in here are trained with this huge amount of samples. It presents two main handicaps:

- First, the high computational cost. Especially when dealing with high-resolution images. The model has to take into account this limitation since we expect them to work in a reasonable amount of time. Another common approach is downsampling the image (usually by a power of 2), which in some cases could help to speed up and improve (the samples are simpler) the rest of the stages.
- There are a lot of redundant data. The stains, stroke are defined by a set of pixels (non-individual pixels). Hundreds or thousands of pixels could determine a shape, background pattern, stains or bleed-through artifacts. Millions of samples do not imply millions of different image patterns. It is worth to mention that DIB is an unbalanced task of two classes where only 5% to 15% of the whole image is part of the foreground [Al-Haddad et al. 2000; Pastor-Pellicer, F. Zamora-Martínez, et al. 2013]. Most of the pixels are background, and they do not present a big challenge to mark them as its class.

If we have an image with $1,2Mpx$ ($1\,200\,000$ samples), which is a medium size image, for training, it means that the fitting function has been trained over those million samples. However, it has only seen one image, and it is hard to generalize to other types of images. Summarizing, we have to use our model to classify millions of samples and be able to generalize and work efficiently with the expected data. Downsampling, as stated, can improve the performance significantly, although it is desirable to work with full resolution images to get more accurate results.

4.2 Cleaning and enhancement as a probability pixel estimation problem

We have been referring to the analyzed task as cleaning and enhancement, but also we used the term binarization. Indeed, we are using these two concepts for the same tasks. Binarization marks the foreground of the image, and it also implies cleaning since we remove dirty backgrounds. It is clear that natural background must be removed (set to 0) and main text should be marked as foreground. But there are some particular cases where we have to agree:

- *If a stroke or part of it is left should it be marked as foreground?*
- *Or if we have a terrible stain in the image or seal. Should it be removed? What about decorations?*
- *What happens with the border of the strokes. If we have high-resolution images the border frontiers between text and non-text could be not clear.*

For example, a subtask of DIB is the text/non-text classification problem, in this case, we mark as foreground only the text segments (avoiding decorations).

Cleaning and enhancement do not imply binarization, that is, they could work at gray level scale. Traditionally, the output of image cleaning is a binarized image where black pixels mean the presence of ink in this region. Nevertheless, since many pre-processing techniques can also deal with gray level images, it is possible to consider the gray level of cleaned image pixels as the probability of ink. Thus, cleaned images shall not be regarded as arbitrary grayscale images but, rather, as a soft estimation of a black and white image which tries to represent, in a limited resolution, the set of ideal strokes. Gray values are a way to account for the probability of picking a black subpixel in this pixel, so intermediate gray values are expected to be found in the borders of strokes. This idea has a correspondence with the desirable anti-aliasing property of geometrical transformations applied in most typical pre-processing stages such as the correction of the skew of the page, the slope of the words in the lines and the slant of the strokes.

This problem can be considered as the joint estimation of the probability of finding ink in pixel areas, as the classification of pixels into two classes or as the retrieval of ink areas in the whole document.

Regarding ANN models, they can be trained using a binary ground truth and then generate a gray scale cleaned image with the soft output of the classifier. We can threshold this value at any time. Also, the model could also be trained with grayscales images, since we backpropagate the output value when training.

4.3 Ground Truth Generation

Our models require supervised and well-labeled data to learn successful models. In particular, DIB evaluation is usually computed at pixel level (Section 4.4), and it requires an accurate ground truth, with the inherent complexity of data supervision at this detail level. Generating synthetic data for training and evaluating document image processing systems is a topic that has been widely addressed in recent years [Baird 2007; Kieu, Visani, Journet, Mullot, et al. 2013; Varga et al. 2003; Zi et al. 2004]. To overcome this issue, there are several techniques to generate useful ground truths. We found two main strategies: *denoising* and *noising* (and the combination of both). The former strategy starts with a dirty image and creates its ground truth by cleaning it, while the later takes a clean image and then a noisy background and artifacts are added. On the one hand, the first approach has the advantage of using a real dirty image, but on the contrary, the supervised cleaning procedure could be harder. The second method has the challenge of finding a realistic noisy model.

Some of the denoising methods, besides, are harder to supervise and require human effort, especially at text boundary pixels. This problem become critical when supervising high-resolution images where it is easy to have ambiguities in the edge pixels

4.3.1 Denoising supervision

Several approaches to remove noise from a picture can be adopted to obtain its clean ground truth.

Image Manual pixel segmentation It relies on removing noise by a human expert, usually assisted by a particular software. This method is discouraged and never used from scratch. Since it requires lot of human effort and can be partially automated as we will see.

Combination of several methods and parameter tuning In this case, the human expert could use some of the conventional binarization/cleaning processes as a starting point. The expert could even stack several of the most reliable approaches, to get better results and ease the final manual correction step. For instance, a pipeline for an image could be removing the background by using a median filter, for example, apply Sauvola filter and correct mistakes by the human expert with the help of an assistance tool.

Use Layout information to extract the foreground One can take advantage of the layout definition, and text line ground truth data for text documents since the foreground will be within the regions marked as text. This approach extracts the text regions and then applies the cleaning procedures only on these areas and marks the rest of the page as background.

For instance, if we wanted to extract the foreground text the followed procedure would be:

1. Cut the lines using the text line definition. (Text lines are defined by regions or polygons).
2. The text line areas are cleaned using one of the previously mentioned techniques.
3. The rest of non-text features are marked as background.

Bootstrap denoising This method relies on semi-supervised methods which improve in each iteration with newly cleaned data. First, one of the previous approaches could be used to get a small supervised subset, to be utilized in the first iteration. Then, an iterative training procedure, cleaning a larger set of images, manual supervision of the mistakes and retraining with the bigger set, is followed, until all the training data is supervised or a convergence criterion is reached.

Thus, a common procedure could be described as follows[1]:

[1]We marked with an apostrophe (X') the sets that have been cleaned or corrected by an expert. Also, we talk about a model θ, we generalize for any machine learning model, but in practice, they correspond to ANN.

1. A small set of images (or patches) A_0 are cleaned by a baseline method.
2. Manual correction of possibles mistakes is performed in the clean set: A'_0.
3. A model θ_0 is trained with the previously supervised images A'_0.
4. The model θ_0 is used to generate a new larger set of cleaned images A_1.
5. This new set is manually corrected (A'_1), and it is used to train a new model θ_1.
6. Iterate between the steps 4 and 5, until obtaining the desired amount of supervised data.

In this method, the first image corrections are more costly, since the model is less accurate. The more data the model has seen, the better results while less supervision effort is required.

4.3.2 Noising methods

Using synthetic data or synthetically degraded data has many advantages over human supervision including rapid generation of datasets at lower cost, control of degradation level, and fit testing of the same underlying document content with different corruption methods [Baird 2007; Kieu, Visani, Journet, Mullot, et al. 2013; Varga et al. 2003; Zi et al. 2004]. The main idea is to take a clean image as the ground truth and apply several distortions and noise on top of it. We are not going into details about noising methods. Nevertheless, it is worth to remark that there are different modes.

When using ML based approaches like ANNs, it becomes useful to know how the ground truth has been obtained. For instance, if the ground truth has been generated only using heuristics methods, there is a risk that the model does not generalize enough to a different real noise.

4.4 Measures and evaluation

Before introducing our developed methods, we would like to discuss the different measures for evaluating the DIB task. As shown in Figure 4.1, the cleaning and enhancement stage is the first (or one of the initials) of the OCR/HWR and DIA pipeline. One could assess the goodness of this stage by checking its impact in all the following stages, e.g. checking the WER after the full the decoding process. Since there are lots of steps between the cleaning and decoding tasks, it is sometimes difficult to understand the real impact of each of the processes in the overall workflow. The basic sceneario consists in keep-

ing unchanged the rest of stages and check how the final results variate respect the new inputs. Another proper comparison of the cleaning/binarization methods is to test the performance in the intermediate stages, since this step has a direct influence in layout and text line extraction stages. But when the ground truth is available one could directly compare both images (predicted and ground truth) and apply one of metrics discussed below. DIB contests perform their evaluation as well by comparing the cleaned image (predicted) with the corresponding ground truth.

4.4.1 Subjective evaluation

We have discussed so far the challenges and difficulties to obtain a reliable ground truth. Indeed, it is not unusual not having it in many cases. Thus, it would be necessary to apply some indirect evaluation instead, like the impact in next stages as discussed.

When it is not possible to apply these indirect metrics, we can rely on a subjective evaluation instead, meaning observations like: "The images look clean, our algorithm is quite good". This is an extreme case, but it could be performed in a more normalized way by asking experts to assess to which extent the pages, regions or figure/graphics have been properly cleaned or even, adding a subjective score. Since our evaluations are based on ground truth/prediction comparisons, we will not go into details about subjective methods, but if the lecturer is interested in the topic, we recommend to read [Ntirogiannis, B. B. Gatos, et al. 2013; Trier et al. 1995].

4.4.2 Objective evaluation

Precision and Recall and F-Measure DIB is mainly a classification problem between two classes: foreground and background. Most of the pixels belong to the background class, and indeed, usually these pixels are easy to classify unless they are part of a stain or bleed-through ink or text frontiers. And of course, it will be more important to classify the foreground pixels correctly since, in other case, we could lose useful information. For this purpose, the F-Measure (FM) is one of the most used metrics for DIB evaluation. In this case, we define the foreground class as our relevant class, so the problem is formulated as a retrieval problem where the foreground pixels must be recovered.

		Ground Truth	
		Foreground	Background
Predicted	Foreground	True Positives	False Positives
	Background	False Negatives	True Negatives

Table 4.1: Relevant/non-relevant classification classes.

FM is the harmonic mean of *precision* and *recall* which are defined as follows:

$$\text{FM} = \frac{2 \cdot precision \cdot recall}{precision + recall} \ . \tag{4.1}$$

- Precision (also called positive predictive value) is the fraction of well-classified foreground predicted pixels.

$$precision = \frac{tp}{tp + fp} \ , \tag{4.2}$$

  where tp are True Positives and fp False Positives.

- Recall (also known as sensitivity) is the fraction of ground truth foreground correctly.

$$recall = \frac{tp}{tp + fn} \ , \tag{4.3}$$

  fn stands for False Negatives.

Soft F-Measure The pure FM based evaluation takes the classification as a binary class. If the classifier outputs a score (or probability) for each pixel when computing the FM, a threshold has to set to classify the sample in any of the four cases (Table 4.1). We could generalize the precision/recall values to take into account probability values between 0 and 1. The computation and also the adaptation for using this soft measure for training our nets are explained in detail in section 8.

Accuracy We have introduced FM as a metric that represents better the capabilities of the evaluated method since it takes into account the kind of errors committed. *Accuracy* takes the number of well-classified patterns by the total number of samples without relying in the nature of the missclassified patterns.

$$\text{accuracy} = \frac{hits}{n}. \tag{4.4}$$

Being our task highly unbalanced, it is easy to get high accuracy just by classifying all the pixels to the most probably class. To overcome this issue one could take the accuracy as the average of background and foreground accuracy:

$$\begin{aligned} \text{accuracy'} &= 0.5\,\frac{tp}{tp+fp} + 0.5\,\frac{tn}{tn+fn} = \\ &= 0.5\,\frac{\text{fg\_pixels\_well\_classified}}{\text{total\_fg\_pixels}} + 0.5\,\frac{\text{bg\_ pixels\_well\_classified}}{\text{total\_bg\_pixels}}\ . \end{aligned} \tag{4.5}$$

Mean Squared Error (MSE) It has the advantage that can work with continous values (grayscale or probability outputs).

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(T_i - Y_i)^2\ . \tag{4.6}$$

Cross-Entropy (CE) error It is similar than the MSE but more common in classification tasks.

$$\text{ce} = \frac{1}{n}\sum T \log Y\ . \tag{4.7}$$

Peak Signal to Noise Ratio (PSNR) It uses another scale for the MSE:

$$10 \cdot \log_{10}(\frac{1}{\sqrt{(MSE)}})\ . \tag{4.8}$$

Since most of the pixels, particularly the vast majority of background pixels, are well classified, the MSE tends to be very low. PSNR uses a log scale to get more comparable results (the larger, the better).

Pseudo F-Measure. This metric was introduced by [Ntirogiannis, B. B. Gatos, et al. 2013] and citing its definition the DIBCO final reports:

> Pseudo Recall/Precision metrics use distance weights on the contour of the ground truth characters. In the case of pseudo-Recall, the weights of the foreground (ground truth) are normalized according to the local stroke width. Those weights are delimited between $[0, 1]$. In the case of pseudoPrecision, the weights are constrained to an area that expands to the GT background taking into account the stroke width of the nearest ground truth component. Inside this area, the weights are greater than one (generally delimited between $(1, 2]$) while outside this area they are equal to one.

Unlike previous metrics, it is required to compute the strokes of the text, a first skeletonization algorithm usually suffices. But it could lead to some ambiguities regarding the skeletonization algorithm applied.

Geometric-mean pixel Accuracy This metric deals with the problem of unbalanced datasets. It takes the proportion of foreground and background pixels between the predicted and ground truth [Paredes et al. 2010].

$$GA = \sqrt{\frac{b}{B} \cdot \frac{w}{W}}. \tag{4.9}$$

Where w is the number of background pixels in the prediction (white), W stands for background pixels in the ground truth (white) and b and B are the respective values for the foreground (black).

4.5 Connectionist methods

There are many different ways of appliying ANNs (and deep learning) to DIB. As we already stated, we have one-to-one pixel correspondences between the dirty image and its denoised version. In this case, traditional connectionist methods run over the image treating each pixel as an individual sample [Marinai et al. 2005]. When treating each sample or pixel, we take into account its normalized value, including also other features or contextual information related to the pixel, like a neighborhood window or its histogram. Thus, ANNs run over the image using a sliding window for extracting the input features, while the output corresponds to the cleaned pixel. In our case, we apply the

pixel labeling techniques described in Chapter 3.7. This is considered a convolution over the pixels of the image since the same weights are applied to distinct patches of the image, but we will refer to them as MLPs.

When tuning a sliding window ANN we can split the model hyper-parameters in two: parameters regarding to the input of the net, and parameters related to its training. The former parameters correspond to the input variability of the samples: input window size, extracted features, transformations, and deformations. For the later hyper-parameters we have, for example, the ones required by the optimizer (e.g in gradient desecent, *learning rate, momentum*); and regularization which we have already discussed about them in chapter *weight decay, dropout*.

In this section we will analyze several ANN approaches used for the current task. The classical approach relies on the utilization of a sliding window which is fed to an MLP. Then we explore the use of extended features as input of the classifier (Section 4.5.1), MLPs are then replaced by CNNs (Section 4.5.3). Our final approach treats the image as a 2D sequences which are computed by RNNs, in this case MDLSTM (Section 4.5.4). The primary purpose is to check the suitability of the explored methods for the current task.

4.5.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) receives a raw input image that is centered on the pixel to be cleaned. Each pixel is seen by the net as an independent sample, since the net does not keep any internal state between samples (Figure 4.2).

This approach is the simplest method explored and initially, we had to setup the contextual window which our classifier receives, i.e. the set of pixels that are fed into the MLP for each input sample (pixel). Given a n-neighboring window, the final input size of the net consists on $(n * 2 + 1)^2$ values.

4.5.2 Multilayer Perceptron with additional features

In the basic MLP, the net only receives contextual information from the centered pixel. The bigger the window size, the more information the net receives. However, the input size grows quadratically with the neighborhood value (n). The complexity of the model (weights) increases drastically with the scale of the input window. Thus, it makes sense to consider the inclusion of additional features from a larger context (Figure 4.3).

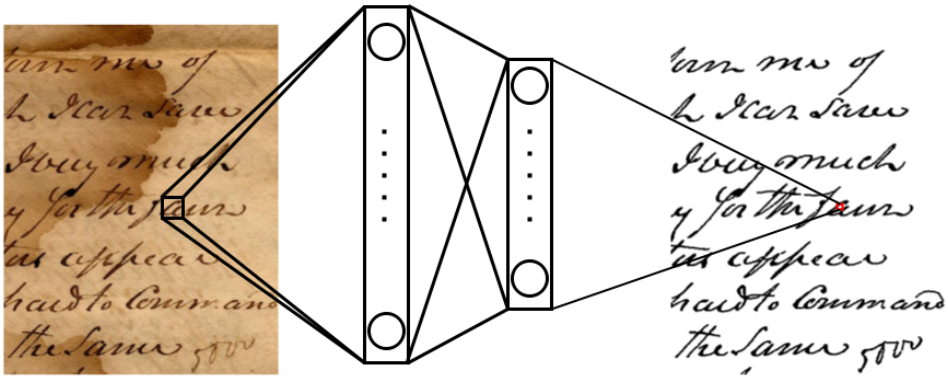


Figure 4.2: The input of the MLP is a centered window on the pixel to de-noise. The output is one single value with the cleaned pixel.

Our MLP using extended features is a refinement of the previous model which adds more detail from the input. The explored features are:

Features	N. of parameters	Description
Input window	$(2n+1)^2$	Input squared window of neighborhood n.
Window Histogram	l	The histogram values of a window of size wh and range l.
Horizontal Histogram	hl	The horizontal histogram values of the pixel column and hn neighborhood columns, and range hl.
Vertical Histogram	wl	The vertical histogram values of the pixel row and wn neighborhood columns of range wl.
Median Filter	1	Median filter is an estimation of the background. It is estimate given a window of radius r.

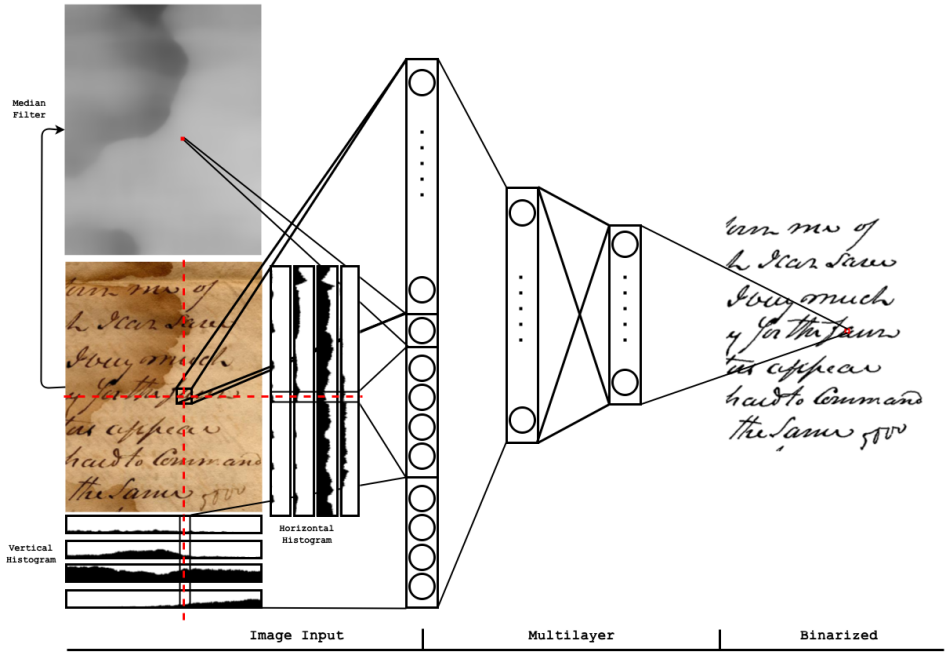


Figure 4.3: New extra features are computed for the centered pixel. The net receives a raw contextual input, then 4 normalized histogram values ($0-63$, $64-127$, 128-191, 192-255) for the vertical and horizontal axes.

4.5.3 Convolutional Neural Networks

The next explored model involves the use of CNNs. The image is treated as one 2-dimensional input map (grayscale, in *RGB* the input would consist of 3 input maps). Therefore, it is possible to apply 2D convolutions for feature extraction. A set of convolution-activation-pooling transformations are applied to the input map(s) to extract a new set of features. In our experimentation, only grayscale images are used, performing a conversion from the *RGB* images when necessary.

Analog to the MLP with extended features, our goal here is to find a useful set of features for classifying each pixel. The main advantage relies on the use of automatic ML procedures to extract those features. The overall procedure is the same than the previous models: we use a sliding window over the image but, in this case, the MLP is replaced by a CNN (with the corresponding dense and classification layers after the convolutions). Thus, the CNN receives a

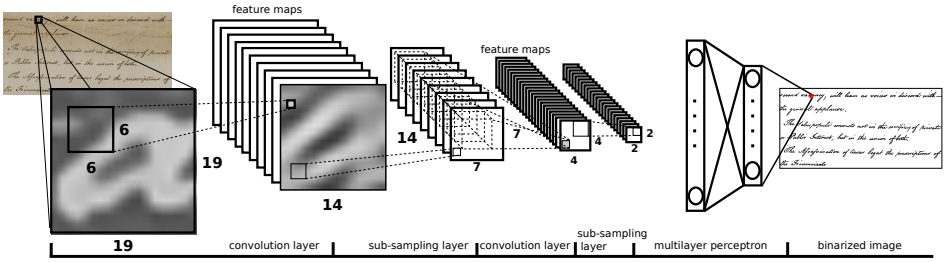


Figure 4.4: The diagram shows the CNN composed of two sets of convolution and sub-sampling layers, and finally, an MLP with 2 hidden layers followed by a single output neuron. The model estimates the value of the cleaned image.

raw input window of the image in order to compute the predicted value of the current pixel.

There are several advantages when using CNNs instead of MLPs: convolutional kernels operate on a smaller scale, and each one shares its weights at different positions on the input window, which reduces the number of parameters decreasing the possibilities of overfitting and improving generalization. When using a sliding window nearby, pixels should have a significant number of features in common since they share the major part of the overlapped window. Thus, two consecutive input windows that look pretty similar have an entirely different representation of the input feature vector because of the window translation. Unlike MLPs, the problem is handled better by CNNs because they maintain the 2D structure of the image and then the kernels can extract similar features from contiguous inputs. Also, max-pooling layers reduce the computational cost and provide translation invariance to the model. Therefore, with this approach, a combination of convolutional and pooling operators should be able to extract more significant features than the traditional MLPs. Unfortunately, these topologies include a full set of new parameters to tune: the number of convolutions, the kernel sizes, poolings and so on. The main challenge of this work is to find a suitable and working CNN topology that could be applied successfully to the current task.

We will explore different setups not only to validate the suitability of CNN empirically for DIB but also to provide some insight relating adequate optimal window sizes and topologies to obtain the best trade-off between binarization quality and computational cost.

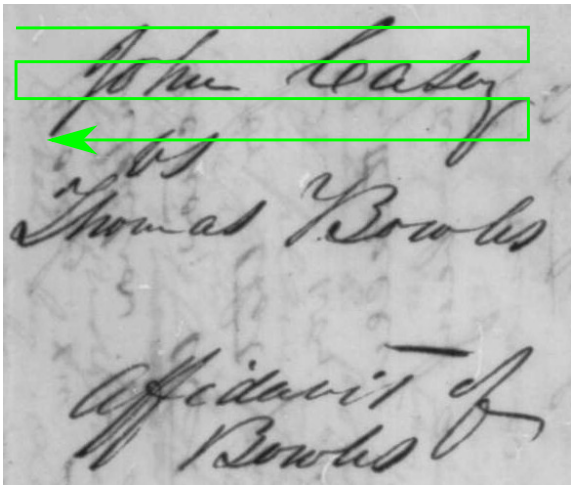


Figure 4.5: 1D RNN for Document Image Binarization. The pixels are scanned row by row in a continuous path.

4.5.4 MultiDirectional Long Short Term Memories

Our last approach is the use of RNNs and particularly, MultiDirectional Recurrent Neural Networks (MDRNNs). RNNs include feedback (or recurrent) connections in their hidden layers and they can deal with arbitrary sequences. For each new sample of the sequence, a temporary state of previous steps is maintained.

As explained in Chapter 3, it is possible to apply 2D-RNN for image labeling. Like with CNNs, the image is seen as a 2D sequence. An alternative approach is to enumerate the pixels as a 1D sequence running over the rows of the images (row major) or the columns (column major). Nevertheless, this is a naive approach since the 1D relations between adjacent pixel do not make any sense when jumping from one row to the next one, and also the 1D hidden state covers a very thin context. These issues could could be softened by having a continuous path from the start point (top-left corner) to the last pixel (bottom-right) as depicted in Figure 4.5. But, in any case, it is desirable to jump directly to the 2D sequences.

In our work we have used LSTM cells modified to keep 2-dimensional context introduced by [Graves, Fernández, and Schmidhuber 2007]. Also, the recurrence is done on the 4 possible orientations as seen in figure 4.6. The output value is extracted as a combination of this four orientations. Thus, when the

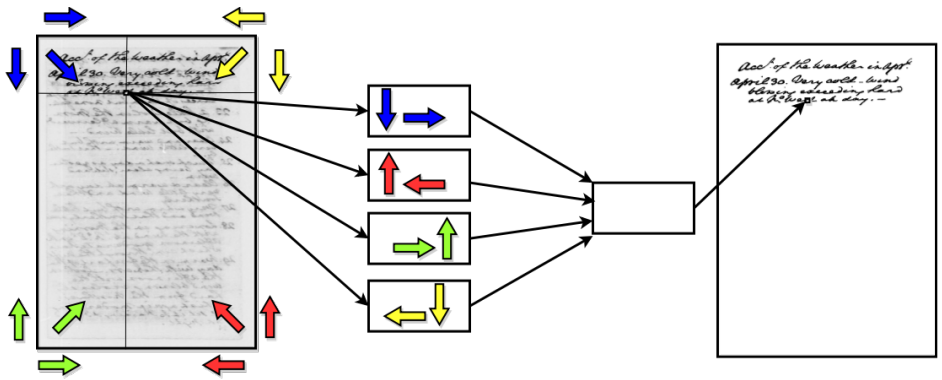


Figure 4.6: Multidirectional Long Short Term Memories: each of the 4 LSTM has seen different context.

value of a pixel sample is computed, each of the 4 recurrent neural networks has the information available from the 4 possible directions (Figure 4.6).

Only one hidden layer has been used for each MDLSTM in our settings (plus the combination layer).

4.5.5 Analytic cost

It is interesting to know the capabilities of the connectionist models presented but also the number of weights (parameters) learned by each one. In this section, for each of the proposed models we define the number of weights used per model as well as the analytic cost of computing one pattern, that is the cost of computing one pattern (i.e. in order to clean one pixel) which usually corresponds with a forward pass of an ANN. In the general case, calculating the values of one layer involves the products of the weights by the activation of the previous layer (or input), the addition of the bias and finally, the activation function. For simplicity, only the number of products (weights) executed in a forward pass are taken into account, since activation cost function and bias are linear with the number of neurons[2]. Indeed, we could make use of fast libraries that compute matrix operations in a very efficient way. We provide the number of weight products applied without relying upon other optimization like batch processing, or other parallelism techniques (*BLAS* or *CUDA*).

[2]In some other problems the activation function computation it is not negligible, for example, the case of a softmax activation with large outputs which involves the computation of the normalization constants. In our case we have only one logistic neuron as output

Multilayer Perceptron The cost of computing one pixel, in this case, corresponds to a forward pass over the fully connected layers. That is the number of weights learned by the net.

MultiDirectional Long Short Term Memories In RNNs (such as MDLSTM) new recursive weights within hidden layers are added. Assuming that each pattern is classified in each (time) step, the cost of computing one sample involves a forward pass with the current inputs and the recursive connections. Bidirectional and Multidirectional cases add temporal (or recursive) dependencies from different directions and dimensions. Since the predicted value is the combination of several layers, to process one (particular) sample it is needed to apply several forward passes until arrive to the desired pixel. For the 2-dimensial case, even though all pixels are calculated by using the 4 full scanning directions, we indicate the amortized cost of computing one pixel for a fair comparison with the other techniques. As described in Equation 4.10, where w is the number of weights for each of the four LSTM hidden layers, HW corresponds to the size of the sequence, and the 4 value comes from the different directions. There is one weight added corresponding to the bias of the final combination layer. N represents the number of cells in each LSTM layer. The LSTM has more weights than a normal neuron as shown in Figure 4.7.

$$\begin{aligned} \text{cost} &= \frac{wHW}{HW} + 1 = 4w + 1 \ . \\ w &= 10N^2 + 11N + 1. \ . \end{aligned} \tag{4.10}$$

Convolutional Neural Networks (CNNs) A convolution layer can by defined by the weights of the kernels that are applied to the several parts of the input ($H \times W$). A $x \times y$ kernel is applied to the input map to generate $(W - x + 1) \times (H - y + 1)$ maps. Each convolutional layer applies k_i kernels to the previous k_{i-1} maps generating k new maps. Following this procedure, equation 4.11 defines the number of weights forwarded in a convolution.

$$\begin{aligned} \text{cost} &= k_i \times \#\text{convs} \times \text{conv\_cost} \ . \\ \#\text{convs} &= (w - x + 1) \times (H - y + 1) \ . \\ \text{conv\_cost} &= k_{i-1} \times (x \times y) \ . \end{aligned} \tag{4.11}$$

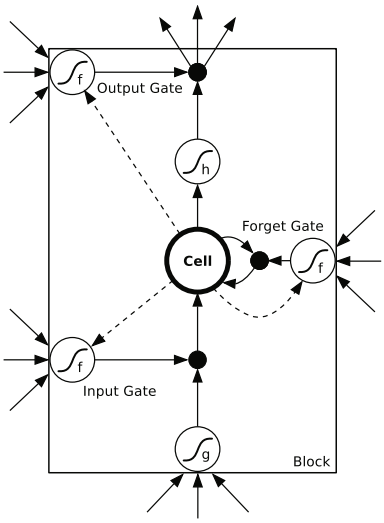


Figure 4.7: Long Short Term Memories (LSTM) memory block. The dotted connections are delayed connections over time. The three gates control the content of the cell. (Figure from [Graves 2008] 4.2).

It is worth noting that each kernel is applied to the input image, but we need only to keep the values of the kernels ($K \times x \times y$). So the number of operations involved is high, but the final parameters of the model are reduced.

4.6 Ensembles

We have been diving through different connectionist approaches to DIB, some will work better than others, but it is always desirable to combine them for achieving better results. The combination of different methods tends to improve results. There are many ways of having ensembles of diferent methods [Arruda et al. 2014; Badekas et al. 2007; B. Su et al. 2011]. Following, we present different ways of combining the proposed methods by taking the various soft outputs; that is the probability values estimated by each ANN.

Baseline Ensembles We can apply a set of straightforward combination methods for a set of n outputs:

- **Averaging** The final value is taking as the average value of the different outputs ($\frac{\sum_i^n x_i}{n}$).
- **Voting** The output values are thresholded (if they are not binary yet) and the most voted class is taken. In the case of even number of classifiers, ties are resolved by applying averaging on the different outputs and then threshold that the output value.
- **Max/Min** The output value is taken as the maximum or minimum values (separated approaches). It could work, for example, with models that tend to get false positives. Nevertheless, this indicates that the methods have not been properly trained since they generate not reliable outputs.

Confidence Ensemble Other than the straightforward combinations, the most natural method is to apply a simple procedure based on some confidence criteria. Since the used ANNs could be seen as a probability estimator, we could take advantage of the logistic output neuron to get a confidence value. A net with a high confidence will have the closest value to 0 or 1, and it may have fewer chances to generate an incorrect classification. But this is also dependent on the training procedure so it is not a very reliable indicator, yet we have seen some improvements using it. Following this idea, we have set up an ensemble method based on confidence where the pixel is classified to the method that presents a higher value of confidence. A formal explanation is set in Equation 4.12 and Figure 4.8 illustrates the proposed method.

$$\text{confidence}(x) = |x - 0.5| \ . \tag{4.12}$$

Minimum Error Rate training A smarter combination is to apply a weighted combination of the different outputs. The final output value is calculated as:

$$\sigma_{comb} = \lambda_1\sigma_1 + \cdots + \lambda_n\sigma_n \ . \tag{4.13}$$

The λ weights can be learned from the training/development data, for example by BP. Although we do not have too many parameters to estimate (between 3

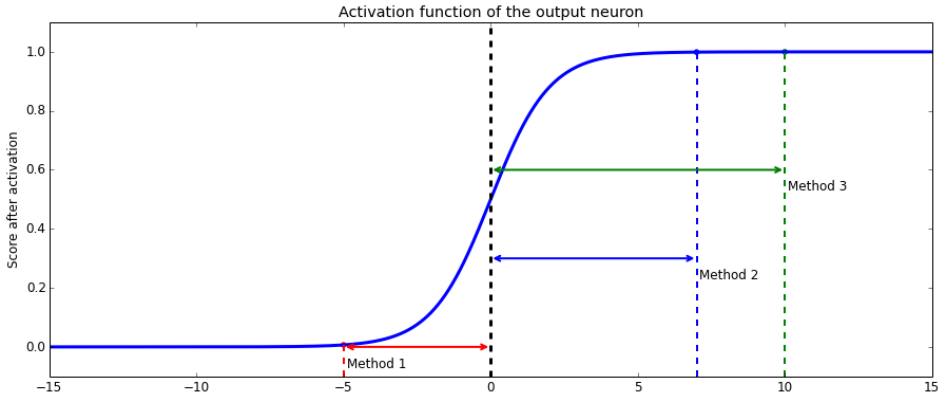


Figure 4.8: Logistic output for a pixel of different methods. The Method 3 has a higher confidence than Method 1 and Method 2. So the pixel is classified as 1 or foreground.

and 4 models) so they could be calculated using a Minimum Error Rate Training (MERT) procedure. Indeed we have used the MERT implementation based on the Simplex algorithm [Nelder et al. 1964]. The weights have been estimated on the development set.

4.7 Experiments and results

In the current chapter, we have seen several connectionist approaches and techniques for addressing the DIB tasks. We want to make a fair comparison with the proposed methods, but also to analyze different topologies, configurations, and apply that to the several corpora. The experimental setup and results have been organized as follows:

- **Insights of CNNs for DIB** It describes the experiments carried out and the configurations explored for having successfully working convolutions on the current task.
- **Comparison of connectionist methods** The different ANNs presented have been applied to the corpora described in Appendix C.1.
- **Ensembles and Regularization** We present the results of the combination methods.

4.7.1 Insights of CNN for Document Image Binarization

We present the procedure carried out in order to get a working CNN for the DIB task. Once we have a better insight into this novel application of CNNs we will be able to perform a more intensive parameter and hyper-parameter search.

Regarding the most appropriate sizes for the window, previous works [Hidalgo et al. 2005; Marinai et al. 2005] has experimentally shown that small windows do not provide enough information whereas large windows add too much variability and lead to problems due to the curse of dimensionality. We will explore a range of windows similar to the ones for the MLP based models. When using MLPs with fully connected layers, the usual topologies comprise one or more hidden layers. Now, we have to take into account a functional set of convolutional and pooling layers. In our initial setups, we limit our nets to up to two sets of convolutions followed by max-pooling layers, and at the end up to two dense layers. The explored parameters are:

- The input window size.
- Size of the kernel of the first convolution.
- Number of kernels in the first convolution.
- Size of the first sub-sampling layer.
- Size of the kernel of the second convolution.
- Number of kernels in the second convolution.
- Size of the second sub-sampling layer.
- Dense layers.

Topology and Parameters Setup To obtain a suitable convolutional topology, we will focus our interest and towards the experimentation to analyze the effect of the number of extracted features from the CNN prior the MLP classifier. This number depends on how many kernels in each convolution and the pooling layers are added. Each kernel tends to learn different characteristics, so the number of kernels is a significant parameter to set up, although increasing this value too much has a direct impact on the computational cost. On the other side, max-pooling layers are used to reduce the dimensionality of the convolved maps.

In our preliminary configuration we have fixed the values of the input window, and also the sizes of the kernels used in both convolutions:

- The sliding window size is fixed to a 9 neighbors leading to a 19×19 window.

- The size of the kernel of the first convolution is set to 6×6 leading to maps of 14×14 real values.
- The size of the first sub-sampling layer is fixed to 2×2 reducing the maps to 7×7, one map for each kernel of the first convolution.
- The sizes of the kernels of the second convolution are set to 4×4, leading to maps of 4×4 before applying the last max-pooling sub-sampling layer.

We have ranged the number of kernels in the first convolution from 10 up to 160, whereas the number of kernels in the second convolution is twice as much the number of the first convolution. Regarding the max-pooling layers, two different settings have been tested for the second sub-sampling layer: 2×2 and 4×4. In the last case, the number of extracted features is drastically reduced (16 values are reduced to 1). In our hyper-parameters tuning process, we have set a final MLP composed of two hidden layers of sizes 32 and 16, respectively. Having such small dense layers, has shown well behaviour in most of the cases; the power of the model comes from previous convolution layers.

An exploration of the parameters depicted previously have been applied to the DIBCO dataset (more information about the data and partition in Appendix C.1). Figure 4.9 shows the FM for development set, in that case, we have increased the number of generated features by increasing the number of kernels. We demonstrated two different approaches with 2×2 and 4×4 max-poolings in the last convolution. For a given a set of F features, the number of kernels after the poolings are $F/8$ and $F/4$, respectively. For instance, a net with 20 kernels in the second convolution layer generates 80 features when using 2×2 layers and 20 features when using the 4×4 max-pooling.

As expected, the more features the better, although the performance slows down around 80 features. The 4×4 max-pooling layer has better performance when using less than 80 features since more kernels are applied. When more than 80 features are used, the 2×2 max-pooling nets perform better probably due to a less extreme sub-sampling.

The kernels learned by the different nets are illustrated in Figure 4.10. They are directly applied to the image to extract features like edges or corners.

For the sake of a better comprehension of the convolution layers, Figure 4.11 illustrates how a sample is computed highlighting the activation values of each layer.

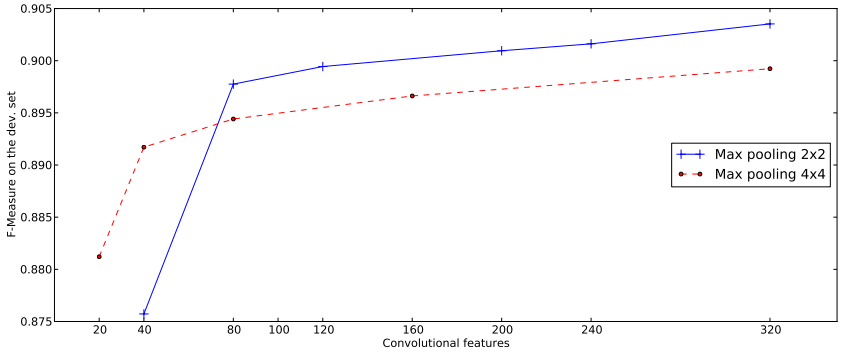


Figure 4.9: FM on the DIBCO development set given the number of features generated by different CNNs (The higher the better).

4.7.2 Comparison of connectionist methods

In this section, we present the experiments performed to analyze and compare all the discussed approaches. Each method has been trained and applied for the corpora described in Appendix C.1. A quick reminder of the analyzed methods:

- Multilayer Perceptron (MLP).
- Multilayer Perceptron (MLP) with extra Features.
- Convolutional Neural Network (CNN).
- MultiDirectional Long Short Term Memories (MDLSTM).

Hyper-parameter tunning For fairer comparison of the proposed models, it is necessary to estimate their parameters (weights) and hyper-parameters (topology, training options) correctly. Thus,the variation between models should be due to the capability of each model and not from the parameter tuning. In Chapter 3.6 we discuss some hyper-parameters tuning procedures, in this case, we have used the random search hyper-parameter optimization. Table 4.2 shows the hyper-parameters sampled, for the MLP, MLP with features, and CNNs: Nevertheless, there a few exceptions of parameters that have not been estimated by this approach: one is the convolutional topology used in the CNN, where convolutional and max-pooling layers have been fixed according to the configurations obtained in previous experiments. We have followed a random search hyper-parameter optimization for each available corpus and each of proposed method (excepting MDLSTM). Figure 4.12 shows the evolution of the error of the best net after several sampling iterations. We could

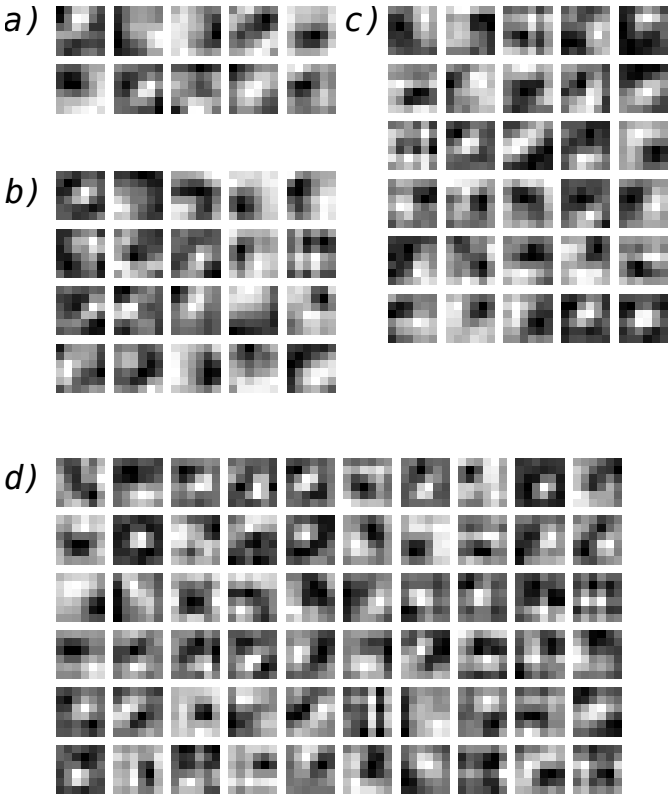


Figure 4.10: Kernels learned from the first convolution of 6×6 for: (a) 10 maps, (b) 20 maps, (c) 30 maps, (d) 60 maps.

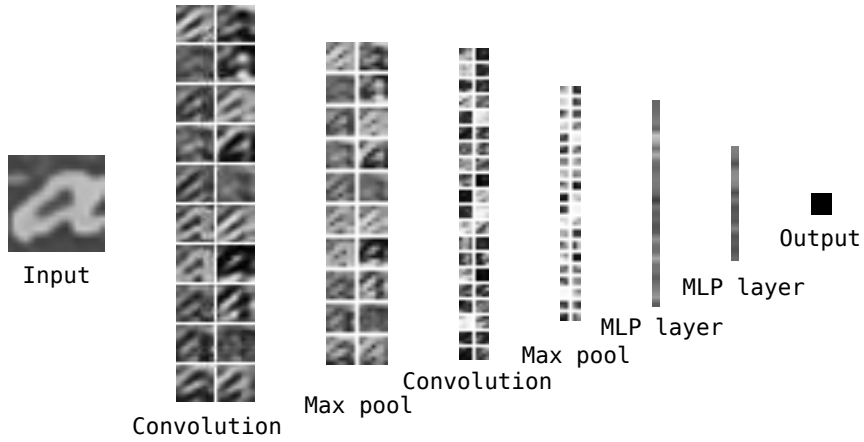


Figure 4.11: Example illustrating the neuron activations of a Convolutional Neural Networks for an input sample (brighter pixels mean higher activations).

Parameter	Distribution	Values	MLP	Feat.	CNN
1st Hidden Layer	uniform	$2^{6\sim9}$	○	○	○
2nd Hidden Layer	uniform	$2^{4\sim7}$	○	○	○
Learning rate	log-uniform	$0.0001 \sim 0.1$	○	○	○
Momentum	log-uniform	$0.0001 \sim 0.1$	○	○	○
Weight decay	uniform	$0, 1e^{-7\sim-4}$	○	○	○
Minibatch	uniform	$2^{0\sim8}$	○	○	○
Input Neighbors	uniform	$4, 6, 8, 10, 12$	○	○	
Median Filter radius	uniform	$\emptyset, 20, 40, 60, 80$		○	
Histogram radius	uniform	$\emptyset, 20, 40, 60, 80$		○	
V. Hist Neighbors	uniform	$\emptyset, 0, 1, 2, 4$		○	
H. Hist Neighbors	uniform	$\emptyset, 0, 1, 2, 4$		○	

Table 4.2: Parameters sampled by the Hyper-parameter Random optimization function.

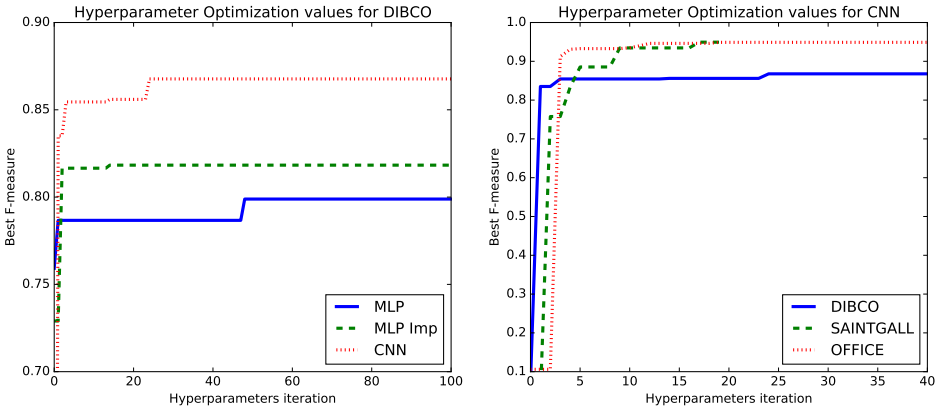


Figure 4.12: Both graphics shows the best result obtained until the current iteration. The left graph shows the error of the different approaches given the DIBCO dataset, while the right shows the error for the different corpora given the CNN method.

see that, after 50 iterations, it is very unlikely to find a better configuration that could improve the overall performance. In our experiments, we kept the tunning procedure running up to 200 iterations.

During the training of the MLP and CNN models, in each training epoch, between $500K$ and $1M$ samples have been taking from a random replacement. Stochastic Gradient Descent with CE error loss has been used for the training. MDLSTMs hyper-parameters, instead, have been tuned by a grid exploration of the learning rate, momentum, and hidden size. In this case, we have split the image in 512×512 patches, and clean each of this independently. For training, random patches are taken as replacement, while for validating and testing, the images are split in non-overlapping patches, and the final output is obtained by joining them. As an example of the best obtained configurations, follwoing the best setups for the DIBCO dataset are detailed:

- **MLP** Two hidden layers of 512×16, learning rate of $1e^{-2}$, momentum of $8e^{-2}$, weight decay of 0, 128 minibatch, and 6 input neighbours leading to an input of 169 values.
- **MLP with Features** Two hidden layers of 64×128, learning rate of $9e^{-3}$, momentum of $7e^{-3}$, weight decay of $1e^{-6}$, 128 minibatch, and 6 input neighbors which made an input of 225 values. Then values are added to the input corresponding to the histogram of the pixels of a window radius of 60 centered at the pixel, plus the median value of this radio. Other 4

are also included corresponding to the horizontal histogram. That makes an input of 178 (169 of the image input, 4 of the window histogram, 1 from the median filter and 4 from the horizontal histogram).

- **CNNs** After analyzing several nets and their performance on the development set as shown in the previous section, we have selected for the final evaluation a CNN showing a good compromise between FM and the computational cost. The chosen one has 10 kernels for the first convolution layer and 20 kernels for the second one. Both max-pooling layers are of size 2×2, leading to 80 extracted features. The following dense layers have 512 and 16 neurons, respectively. The net have been trained using a learning rate of $1e^{-2}$, momentum of $8e^{-3}$, weight decay of 0 and minibatches of 128.

4.7.3 Results

Here we collect results of our methods on the evaluated corpora. Besides, baseline heuristic approaches have been added: Otsu's and Sauvola's. Table 4.3 collects the different measures for all the methods on the tests sets. And the graph in Figure 4.13 illustrates these results. The Table 4.4 shows also the FM for the different sets, Table 4.5 shows a comparison of our best method (CNN) with other used on the DIBCO-2013. And finally, some examples together with the result of these binarization techniques are illustrated in Figure 4.14.

Comparing the performance of the Convolutional Neural Networks approach on the DIBCO dataset with respect to the results reported in the DIBCO-2013 [Pratikakis et al. 2013], the best FM on this competition was 92.70. Although the obtained result (87.74) is far from this position, it goes hand in hand with other competitors.

Ensembles and combinations We have taken the nets that obtained the lower errors on validation and combined following several approaches. Another approach could be to get the best nets obtained while setting the hyperparameters, no matter what kind of classifier. By combining the results from different methods, it is more likely to compensate their mistakes. MDLSTMs, however, performed worse in some cases and we noticed that adding them to the combination degenerate the results. Hence, we have removed them from the ensembles. Table 4.7 shows the FM scores for the different test sets. It includes the score of the 3 methods alone and the several combinations.

		FM	PR	RC	MSE	PSNR
DIBCO	MLP	0.823	0.849	0.859	0.029	16.89
	MLP-Features	0.858	0.862	**0.903**	0.021	18.18
	LSTM	0.782	0.853	0.797	0.039	15.74
	CNN	**0.877**	**0.950**	0.847	0.020	18.91
	Otsu	0.755	0.869	0.757	0.056	15.16
	Sauvola	0.761	0.859	0.750	0.047	14.91
	Ensemble	0.857	0.934	0.836	0.024	18.18
Saint Gall	MLP	0.952	0.931	0.974	0.003	25.17
	MLP-Features	0.958	0.955	0.961	0.003	25.75
	LSTM	0.849	0.843	0.856	0.010	20.12
	CNN	**0.970**	0.965	0.975	0.002	27.22
	Otsu	0.767	0.744	0.793	0.015	18.34
	Sauvola	0.868	0.936	0.810	0.009	20.34
	Ensemble	**0.990**	**0.993**	**0.987**	0.001	32.05
Saint Gall CROP	MLP	0.939	0.940	0.937	0.005	22.80
	MLP-Features	0.947	0.951	0.942	0.005	23.39
	LSTM	0.812	0.953	0.708	0.019	17.25
	CNN	**0.970**	**0.965**	**0.975**	0.002	27.22
	Otsu	0.807	0.959	0.698	0.020	17.09
	Sauvola	0.886	0.936	0.841	0.010	19.86
	Ensemble	0.969	0.967	0.972	0.003	25.85
NOISY OFFICE	MLP	**0.976**	0.975	0.976	0.007	22.50
	MLP-Features	0.974	0.987	0.961	0.007	22.05
	LSTM	0.922	0.967	0.893	0.025	18.55
	CNN	0.970	0.997	0.945	0.008	21.26
	Otsu	0.851	0.970	0.814	0.074	17.03
	Sauvola	0.952	0.978	0.927	0.013	19.01
	Ensemble	**0.997**	**1.000**	**0.994**	0.001	31.37

Table 4.3: Results on the different corpora. All the evaluated metrics but MSE are better if the values are higher.

	DIBCO		
	Train	Val	Test
MLP	0.86	0.85	0.82
MLP-Features	0.88	0.89	0.86
LSTM	0.80	0.77	0.78
CNN	0.93	0.90	0.88
Otsu	0.82	0.79	0.76
Sauvola	0.80	0.81	0.76

	Saint Gall		
	Train	Val	Test
MLP	0.95	0.96	0.95
MLP-Features	0.96	0.96	0.96
LSTM	0.85	0.85	0.85
CNN	0.97	0.97	0.97
Otsu	0.74	0.75	0.77
Sauvola	0.88	0.87	0.86

	Saint Gall CROP		
	Train	Val	Test
MLP	0.94	0.94	0.94
MLP-Features	0.95	0.95	0.94
LSTM	0.82	0.81	0.81
CNN	0.97	0.97	0.97
Otsu	0.81	0.80	0.81
Sauvola	0.89	0.88	0.89

	NOISY OFFICE		
	Train	Val	Test
MLP	0.98	0.98	0.98
MLP-Features	0.98	0.98	0.97
LSTM	0.96	0.94	0.92
CNN	0.97	0.97	0.96
Otsu	0.941	0.92	0.85
Sauvola	0.96	0.96	0.95

Table 4.4: Overall FM for the methods and the different datasets: training, validation and test (the higher the better).

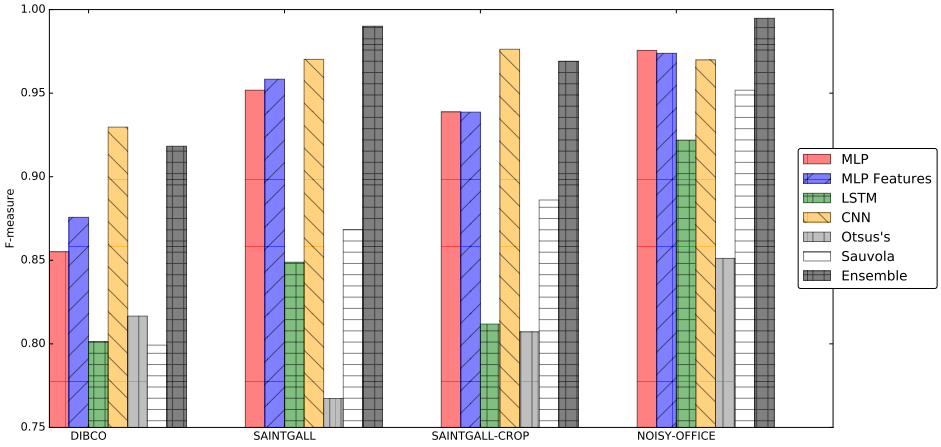


Figure 4.13: The bars shows the F-Measure (FM) that are obtained for all the methods and the different datasets presented.

	Method	FM	Fps	PSNR	DRD
15*b*	Su, Lu	92.12	94.19	20.68	3.10
3	Howe	92.70	93.19	21.29	3.18
5	Moghaddam, Moghaddam	91.81	92.67	20.68	4.02
13	Lelore, Bouchara	91.69	92.16	20.54	3.54
17	Ramirez-Ortegón	90.92	92.82	19.32	3.91
10*c*	Hassaine, Hassaine	89.77	90.36	19.26	4.31
9	Neves, Zanchettin	89.46	89.95	19.05	4.72
11	Roe, A.B Mello	89.05	91.40	18.73	4.36
8*b*	Okamoto, Nakata	88.58	90.81	18.66	4.66
2	Reddy, Chattopadhyay	88.45	88.91	18.66	6.36
	CNN	**87.68**	-	**18.91**	-
4	Yoshida	87.35	91.80	18.34	4, 40
16	Nicolau	83.24	86.59	17.64	6.45
12	Raza	86.16	86.36	17.29	6.51
14	Sehad, Chibani	78.73	86.82	15.25	11.30
1	Djeddi, Labiba	64.62	65.35	11.10	46.09

Table 4.5: Comparison of our best approach with the standings on DIBCO-2013

	DIBCO 2013			Saint Gall		
Method	FM	MSE	PSNR	FM	MSE	PSNR
Otsu	83.94	0.056	16.94	80.71	0.020	17.09
Sauvola	85.02	0.047	16.63	88.68	0.010	19.86
MLP	82.31	0.029	16.89	93.94	0.005	22.80
MLP+Features	85.82	0.021	18.18	94.75	0.005	23.39
CNN	87.74	0.020	18.91	97.02	0.002	27.22

Table 4.6: Performance of CNN on DIBCO and Saint Gall databases.

Figure 4.14: Examples of binarization. First column: Full DIBCO 2013 sample and the obtained binarizations. Second column: DIBCO 2013 image and its binarizations.

Dropout Dropout seems an appropriate technique to apply to some of the ANN used for DIB. For example, in the DIBCO dataset, we have a tremendous variability with the different sets. Indeed, images are quite different in general, contained both handwritten and printed material.

	DIBCO	Saint Gall	ST. CROP	NOISY OFFICE
MLP	0.828	0.952	0.976	0.976
MLP-Features	0.858	0.958	0.974	0.974
CNNs	0.877	0.97	0.970	0.970
Max	0.830	0.959	0.950	0.951
Min	0.863	0.963	0.954	0.983
Avg	0.865	0.982	0.963	0.99
Voting	0.857	0.974	0.955	0.986
Confidence	0.870	**0.990**	0.969	**0.995**
Simplex	**0.879**	0.975	**0.975**	**0.995**

Table 4.7: FM scores for the different combination approaches.

We evaluated also dropout combined with ReLU activation. We applied it to the MLP (without extra features) and the CNN. We have variate the *droprate* on the fully connected layers for both classifiers. In the convolutional layers, we have used a fixed droprate of 0.2 except for the net without dropout. The rest of the parameters is kept untouched after the best configurations obtained previously. In another range of experiments, we tuned the droprate in the convolution layers, but we have seen that dropping activations in there does not improve the overall performance of the net. What is more, higher droprates worse the results. The tendency of the FM error on the DIBCO sets is shown in Figure 4.15. From that graph, one appreciates how in the case of the MLP the dropout helps until some level, while in CNN dropout does not improve the results. We presume that the kernels used by the convolutions generalize better, especially with the max-pool layers, so dropout in the classification layers does not have such improvement.

4.8 Discussion

Let us summarize the methods and the outcomes presented so far.

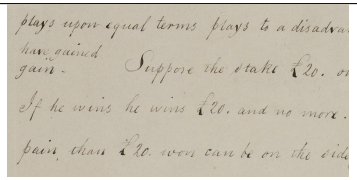
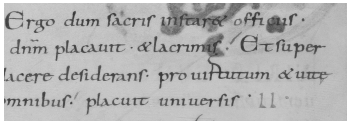
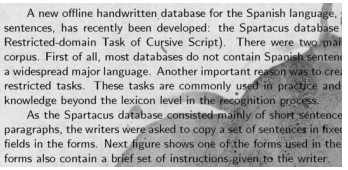
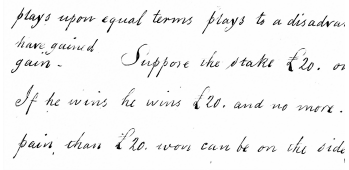
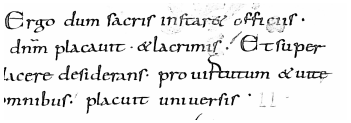
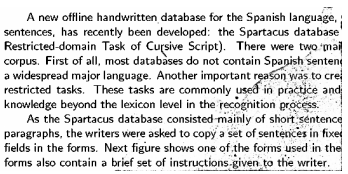
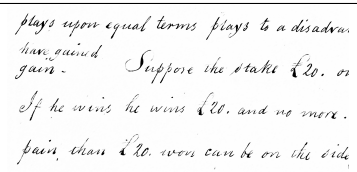
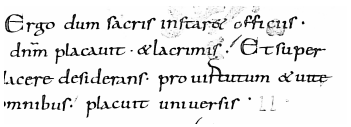
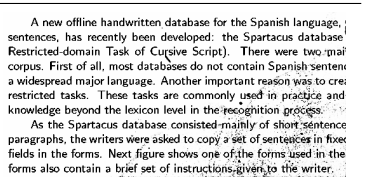
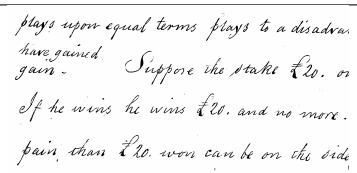
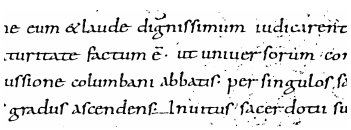
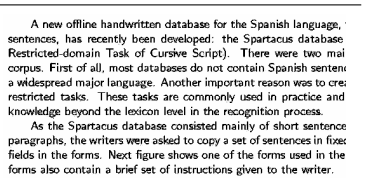
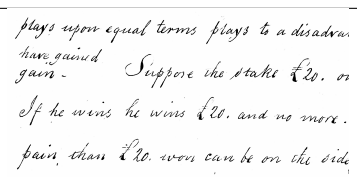
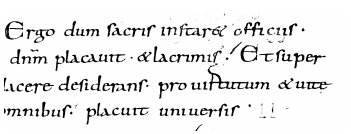
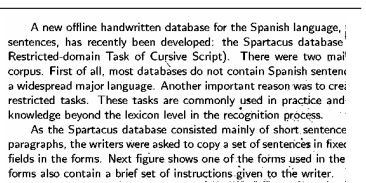
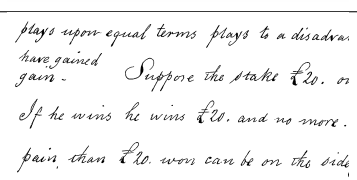
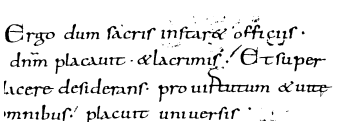
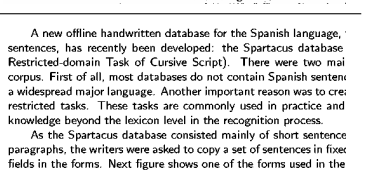
Model	DIBCO	Saint Gall	NOISY OFFICE
Original			
MLP			
Feat			
CNNs			
Ensemble			
GT			

Table 4.8: Illustration of the binarization of the different corpora and several methods.

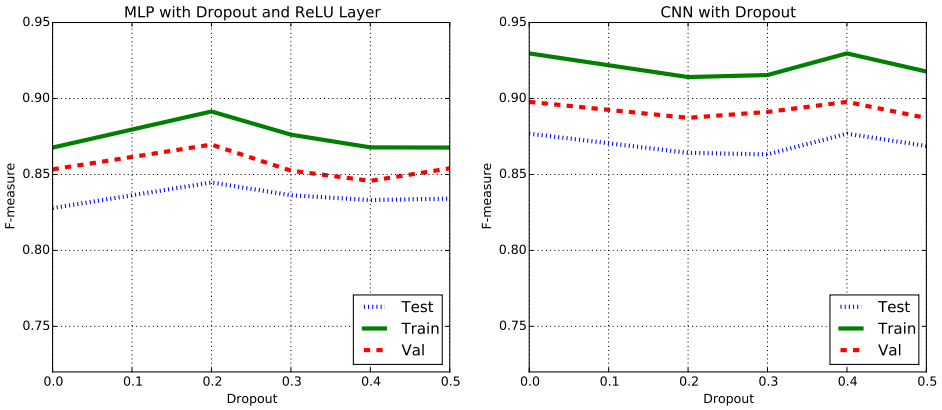


Figure 4.15: FM scores adding different levels of Dropout on the fully connected layers for the DIBCO sets.

Insights of Convolutional Neural Network (CNN) for Document Image Binarization (DIB) Experimental results on the different datasets show that CNNs systematically outperform MLP for this task. The improvement for the Historical IAM Database (Saint Gall and Parzival) seems more prominent than the results on DIBCO. This difference may be due to the fact that the previous collection is more homogeneous than the later. Indeed, it seems that methods based on supervised learning techniques excel in this kind of documents where the font and size of the text, as well as the kind of noise, is more homogeneous along the full collection. The practical interest of the proposed technique is supported by the existence of collections composed of thousands of similar documents. Due to the enormous number of topologies and setting parameters of these models, the study of the influence of the most sensitive parameters and the proposal of working and practical topologies provides a useful insight into the use of ANN for DIB.

It is convenient to obtain a good compromise between binarization quality and computational cost. We have also shown that small kernels are enough to achieve competitive results and empirically validated the effect of the number of such kernels on the overall binarization performance. In fact, increasing this number over a given threshold does not lead to significant improvements.

Comparison of connectionist methods After comparing all the proposed methods with all the analyzed corpora, the first remarkable conclusion is that supervised methods outperform the baselines. The only exception comes with the MDLSTMs that had some problems to work well with the proposed corpora. In this area, we have made several improvements by using some additional pre and post processing that gave us the best result on the DIBCO-2013 test set: 89.82 on FM [Afzal et al. 2015]. Nevertheless, this is not a fair comparison with the rest of the models presented here so that we will keep this out of this discussion.

We have to emphasize that our best net does not give us the best result for the competitive DIBCO set. The results are discrete but not bad as shown in Figure 4.5. This set of images (Appendix C.1) does not give the best scenario for supervised methods. We could expect an improvement of the results if more significant data is fed to our connectionist model.

It is also interesting to note that CNNs have outperformed other previous neural methods, even with the same input. Convolutional layers can extract more reliable features from the image than traditional neurons or other parametrization. It is more remarkable in the DIBCO dataset where we have seen that is tough to improve the performance due to the variability in the images. For counterpart, in the case of the Noisy Office database, which is a very easy set of images, the simplest method based on MLPs obtains a better performance than the other more complex nets.

Coming back to the MDLSTM based model, the results are worse than one could expect. This novel method, in theory, should be able to perform well in this task since it can get a bigger context dependencies in its internal state. We believe that there could be many reasons (or the combination of them) for explaining this issues:

- A very basic setup have been used. We presume that better results could be obtained with more complex architectures like the hierarchical nets and also the combination of convolutional features and MDLSTM.
- The resolution of the images, we use patches of 512×512 which mean that for one pixel we are having a context of $262\,144$ pixels. The 2D structure and the impossibility of reducing the resolution of the images due to the nature of the task constrain the patch sizes.

Combination and ensembles Apart from the straightforward combination, we have seen that the confidence ensembles and the MERT, as shown in Table 4.7 improve the individual results. As seen, the combination of approaches works very well when the combined methods have similar error rate (not necessarily the same kind of mistakes). If this gap is high, the bad result tends to slant the combination. It is worth remarking that in the case of the Saint Gall and Noisy Office corpora, the smart ensembles showed an almost perfect performance. In the cropped version of Saint Gall, the combination works worse than the original since the cropping method is not perfect and there are small artifacts in the borders of the image that add false positives in the final evaluation.

4.9 Summary

We have presented different neural models to approach the DIB task. A thorough search of parameters and topologies has been followed for all of them. Also, we applied successfully CNNs for the current task and explored several configurations to make them work and better than previously connectionist approaches. The publications derived from this research are: [Afzal et al. 2015; Pastor-Pellicer, España-Boquera, F. Zamora-Martínez, et al. 2015].

Chapter 5

Text Line Extraction

Contents

In this chapter we treat the next stage on the Document Image Analysis pipeline: Text Line Extraction (TLE) (Figure 5.1).

As its name states, it consists on delimiting and extracting the text lines of the document for further recognition. The input of the TLE module is a document image (presumably pre-processed). And the output contains the line definitions; these could be just the surrounding areas, but more information such as the type and content of text may be added. Therefore, the primary goal is to extract the single lines that will be used by the transcription engines

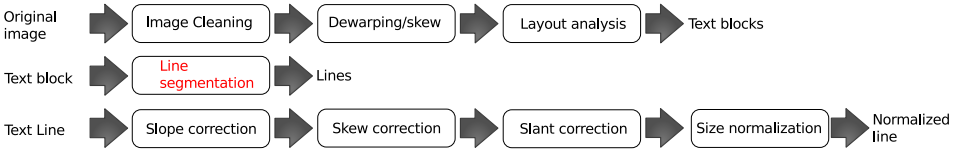


Figure 5.1: Pre-processing steps (Text Line Extraction).

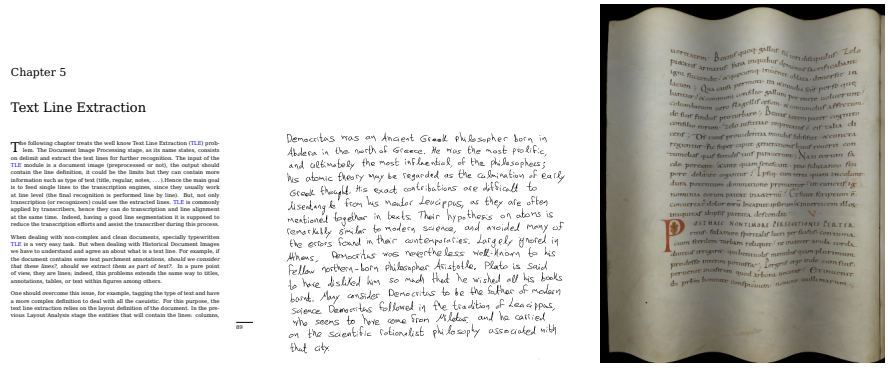


Figure 5.2: Sample of lines from different documents. Each one shows an increasing TLE difficulty. Lines in the first document are clean and well defined, straight forward techniques (histogram or CCs) will suffice to detect and segment them. The document on the middle is clean but it presents some irregularities that require more advanced techniques. The last document shows (artificial) distortions and special glyphs.

since these usually perform the recognition line by line. TLE could also be used to assist transcribers when a reliable recognition engine is not available. When dealing with non-complex and clean documents (specifically typewritten) TLE becomes an easy task. It gets harder, for example, on unconstrained handwritten (Figure5.2-center) or historical distorted documents (Figure5.2-right). Besides, in other scenarios such as Historical Documents, we have to define what is a text line and how it is represented. E.g., if the document contains parchment annotations, *should we consider these as lines?* Indeed, this problem is extended to titles, tables, signatures or even text within figures and decorations. The text type information included in the layout definition could be useful in later stages like TLE.

The entities that contain the lines (columns, titles, text blocks and, annotations) are extracted an conveniently labeled in the Layout Analysis stage. Other elements of the document which do not include text such as decora-

tions, figures or signatures are also detected and labeled. Therefore, the input of the TLE stage is not always a raw image, but a text region extracted and tagged in the previous layout extraction step. Indeed, more accurate and specialized techniques could be applied regarding the class of the area. If we followed a supervised approach, the method would consider as line what the ground truth says that is a line. Our supervised models will learn the features about what lines defined in the ground truth.

There are several scenarios where TLE could be applied, those are not exclusive and, for sure, there will be particular cases that are not contemplated here:

- TLE is part of an end-to-end recognition engine. It could be embedded in the recognition module, used as external web service running, or as a batch tool without (a priori) human supervision.
- It is part of a transcribing assistance tool. The lines are extracted, and then experts transcribe the documents line by line. Another scenario performs a text line segmentation and then iteratively the human expert corrects and improves the line model.
- Evaluation contests used in the academic and research communities for fair comparison of several contributions. In this case, even though the objective is to provide reliable and accurate text line ground truth, the methods are often tuned to get high scores on the evaluated metrics.

For all these cases, a thorough understanding of the ground truth formats and the line definition are required to develop, use and spread the techniques proposed in the following chapter.

We explain in this chapter the TLE task and its applications, and we introduce our proposals for this task. We also include a review of the ground truth formats and their particularities, advantages, and handicaps in the Appendix D.2. We present and discuss the evaluation measures on the proposed tasks (Appendix D.1).

Figure 5.3: Text Reference Lines.

5.1 Ground truth formats and evaluation metrics

In the fulfillment of the proposed approaches, we had to deal with a vast range of formats and different evaluation metrics.

The main problem of TLE evaluation is the line assignment procedure: a one-to-one assignment between these both sets is necessary. The Appendix D.1 describes the evaluation measures applied to our techniques:

- Match Score (MS).
- Pixel Level Hit Rate (PHR).
- Text-Line-Level Detection Rate (TLL-DR).
- Precision, Recall and FM.

and Table 5.1 summarizes the metrics.

It is convenient to know and work with several formats and schemas to ease the interchange and publicity among the research community. In Appendix D.2 we discuss some of the proposed solutions for the *Text Line Ground Truth* definition. We present their advantages and handicaps, and the ideal scenario to use each of them. Besides, one cannot say which definition is better; it depends on the situation and the actual goal. It is also worth to mention the work developed in [Shafait, Keysers, and T. M. Breuel 2006] since they present different evaluation metrics and representation of different DIA tasks.

5.2 Text Reference Lines

In the following sections we introduce our approaches to the TLE task. Our methods follow the idea of tracking the *text reference lines* (and the text line areas/zones delimited by them). An illustration sample showing the reference lines of the text is shown in Figure 5.3. Therefore, our methods are meant to scripts which present these text reference lines. It includes most of the alphabetical scripts, mainly the *Latin* family.

Metric	Level	Description
Computes a line match if the intersection of the foreground pixels between a predicted line (R_i) and the ground truth line (G_j) is higher than a certain threshold.		
DR	Line	Tracks the number of predicted lines that have been correctly assigned (precision).
RA		Evaluates the correctly detected ground truth lines
FM		Trade-off between DR/RA
One-to-one alignment between predicted lines and ground truth is obtained.		
PHR	Line	Number of shared foreground pixels between each of the pairs, normalized by the number of the foreground in the ground truth.
$tll - dr$		Number of well detected lines. A line is claimed to be detected if it shares 90% of the foreground pixels
It treats the predicted/ground truth lines assignment problem as an alignment problem where we can have substitutions, deletions, and insertions.		
tla	line	$Acc_{T_\triangle} = \frac{N-S-D-I}{N}$
$tllpa$	pixel	$Acc_P = \frac{N_P - S_P - D_P - I_P}{N_P}$
$Precision$ $Recall$	line	Generalizes the PHR and TLL-DR to compute the pixel/line precision and recall.
$Precision$ $Recall$	pixel	

Table 5.1: Summary of the different used metrics.

Baseline (or lower baseline)

[Wikipedia] The baseline is the line upon which most letters “sit”.

It is a very reliable indicator for guessing the line orientation since the character sequences sit on it. Indeed, during the slope correction step, we put all the words from a sentence on the same baseline.

Mean line (or upper baseline)

> [Wikipedia] In typography, the mean line, also called the midline, is half the distance from the baseline to the cap height.

Put in other words; it is the line that delimits the upper part of the lower case letters.

Ascenders and Descenders lines

> [Wikipedia] In typography, an ascender is the portion of a minuscule letter in a Latin-derived alphabet that extends above the mean line of a font.

Ascenders and descenders lines delineate the extension of the line on the vertical axis. E.g., the part of a lowercase letter that is taller than the font's x-height.

Capital height defines the upper limits of the capitals letters, this reference line is close to the ascenders line, and sometimes indistinguishable from it.

Besides the text reference lines define the text zones or areas of the text: Main Body Area (MBA), ascenders and, descenders. Note that the MBA contains the part that has the most useful information contained since it comprises the body of the non-capital letters.

Once the text reference lines and the text areas have been introduced, one could observe that this information could be excellent indicative about the positions of the lines along the document. How to detect this reference lines and areas (or zones) and employ them for the final text line segmentation are the two challenges we had to deal when developing our methodologies.

In the next sections, are detailed the supervised approaches for tracking the text reference lines and later we will show how to use this information in order to extract the text lines.

5.3 Text reference line estimation by Local Extrema Points

We followed two main approaches for extracting text lines by means of estimation of text reference lines:

- Computation of text segments by joining text reference lines.
- Detection of the text zones directly on the document image.

Both procedures require the detection of the text reference lines rely on supervised methods (ANNs). In previous works, the research group has successfully used Local Extrema Points (LEPs) for tracking text reference lines in several pre-processing stages, specifically, for slope correction and text line normalization [Espana-Boquera et al. 2011; Gorbe-Moya et al. 2008]. In those works, the reference lines were estimated directly at line level. However, we have generalized it for the whole document.

The underlying idea of obtaining LEPs is to classify them to belong to one of the reference lines (or none). As we will see, most of these extrema are part of the upper and lower character limits, which correspond to one the of the reference lines. Once a LEP is assigned to one of the reference lines, we refer to it as Interest Point (IP) since it has gained knowledge about its function in a text line. Therefore, given one text line, if we join the IP classified to the same reference line we obtain an estimation of the whole reference line.

With regard to TLE, text reference lines, which follow the orientation of the line, could provide useful information to detect and segment the whole text line. These reliable indicators will help to improve the text line segmentation procedure, and what is more, the method is hence more robust to skewed or even distorted documents.

5.3.1 Local Extrema Points

First of all, we need to define what is a LEP. Note that the term LEPs could be misleading, specifically in the computer vision field; we refer to Local Extrema Point (LEP) as the geometric extremes of the ink strokes. In other research areas, extrema points are taken w.r.t its brightness level and not their spatial position. We have used two different approaches for extracting LEPs from input images: one takes the upper and lower contours from the text and extracts

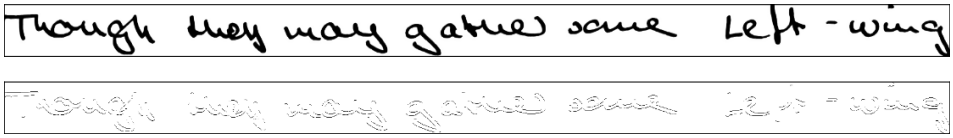


Figure 5.4: Example of the upper and lower contour extracted for one text line.

the extrema in each of the continuous strokes; the second, takes the group edges/contours, groups them and removes the LEP finally from the clusters:

1. **Upper and lower contours** This is the approach used in our original text line normalization techniques. A contour line could be represented as as a continuous function $f(x) = y$ in a 2D plane; the LEPs correspond with the local maxima and minima of the f function. Local maxima values are computed for upper contours and local minima from lower contours (Figure 5.4). The full procedure is detailed in [Gorbe-Moya et al. 2008].

2. **Contour clustering** In this mode, the contours (or edges) are extracted by the Canny edge detector [Canny 1986]. Then, these are grouped by proximity using the DB-SCAN clustering algorithm. Once the contours are grouped by proximity, the LEPs are finally extracted. We define a LEP from the contour as the maximum or minimum point of the n LEPs cluster neighbors.

The reason of using contour clustering is twofold:

- Speeds up the LEP extractor since each subset is computed independently. The main issue is that the clustering algorithm has to deal with a big set of contours points (around $5K$ and $50k$ per page). The clustering procedure could be efficiently computed by using, for example, a *Kd-Tree*.

- It provides an intial first text segment estimation. We will see that in the following stages we will aggregate points to form text segments. The contour clustering provides a first grouping hypothesis.

5.3.2 Interest Points

The next step consists of classifying the LEPs in one of the references lines. The possible categories correspond mainly with the text reference lines:

- Mean Line
- Baseline
- Ascenders line
- Descenders line
- Text (nonbelonging to one of the text reference lines)
- Noise

Note that we have added 2 new classes which cover points not belonging to any of the reference lines. These new classes will distinguish points that are extracted from decorations and stains. We noticed that these non-text points are useful in order to discard non-text areas.

Supervised classification We used a supervised approach to classify LEPs into IPs, for this purpose a supervised ground truth is required. I.e. a set of LEP conveniently labeled that can be used to train our classifiers. The supervision of IP procedure is detailed in Appendix E. Indeed, since the IPs are an artifact that we have introduced to extract the text reference lines, we refer to it as *soft* ground truth. Given a set of well-classified IPs, we train a ANN that receives as an input the contextual information around the points to classify. The output of the classifier is a softmax layer with the 6 classes proposed.

For the classification of IPs we have followed (again) two different approaches: (1) using a *Fish-Eye* transformation, which was the already followed in previous works and (2) applying a CNN model that does not use any previous image transformation.

Fish-Eye Transformation and MLP The input image is distorted using a fish-eye transformation. This consists of a nonuniform image transformation on the horizontal axis. Besides reducing the size of the image, it preserves the resolution of the center part. An input patch (500×200 pixels) centered on the LEP to classify and then downsampled to a 50×30 image which is used as input by the MLP. The fish-eye transformation preserves the information of the central part of the original input, but it comprises the rest of the contextual information as well (Figure 5.5). As depicted in Figure 5.6, the resultant image is flatten and fed to an MLP.

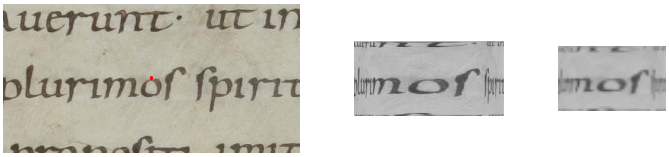


Figure 5.5: Fish eye transformation. The image on the left shows the receptive field around the LEP (red). The middle image shows the surrounding area of the LEP after applying the fisheye transformation (high resolution). The image on the right shows the final input of the MLP after downsampling the fisheye-transformed input window.

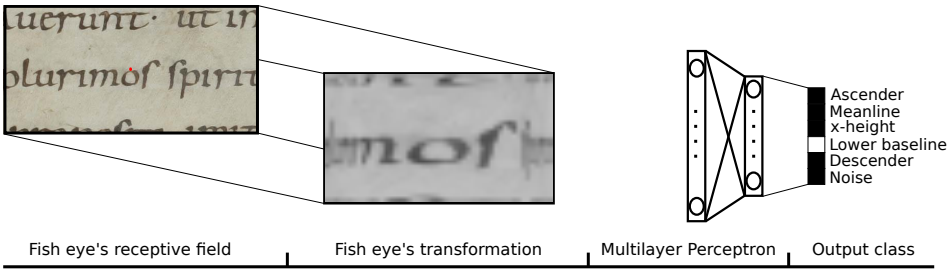


Figure 5.6: IPs classification process by using appplying a fish eye's transformation and an MLP.

Convolutional Neural Network The second approach does not rely on any image transformation or distortions before the ANN. The receptive field of the CNN is pixel window centered around the LEP to classify, as well (Figure 5.7). Then a set of convolutional layers are applied before the final MLP discriminator. The input window is usually smaller than the one in the Fisheye transformation; this can be softened by using downsampling, in our experiments reducing the image by a factor of 2 gave better results.

Any of both methods were good enough for the classification purpose. The classification results were very similar but the difference is clear: the fish-eye transformation does a considerable size downsampling preserving closer and reducing surrounding information, while convolutions, instead, run over all the input region, giving the same importance to center and outer pixels.

Once we have our models trained, we could discrete the soft outputs (argmax) or either propagate them to the next step. We have always tried to take advantage of the ANN estimation by propagating these information we let the

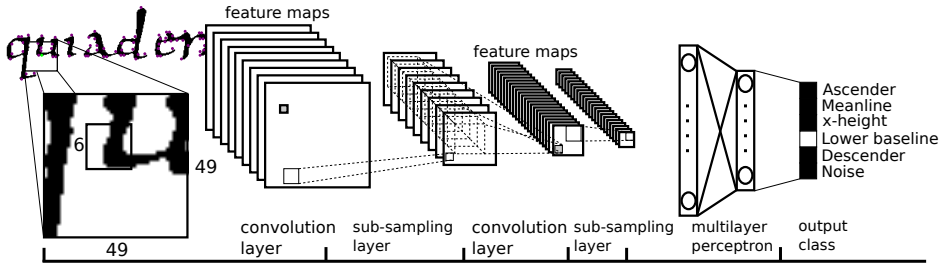


Figure 5.7: Illustration of the LEP classification. The receptive field is a centered window on the point to classify. Then several convolution and sub-sampling layers are applied.IPs

Layer	Type	Kernel Applied	Output
1	input		$1 \times 49 \times 49$
2	convolutional	10 6×6 kernels	$10 \times 44 \times 44$
3	Max-pooling (relu)	2×2	$16 \times 22 \times 22$
4	convolutional	20 4×4 kernels	$20 \times 19 \times 19$
5	Max-pooling (relu)	4×4	$20 \times 4 \times 4$
6	flatten		320 neurons
7	fully connected (relu)		512 neurons
8	fully connected (relu)		256 neurons
9	Output (softma)		6 neurons

Table 5.2: Topology used for classifying the LEP in one of the text reference classes and dirty IPs.

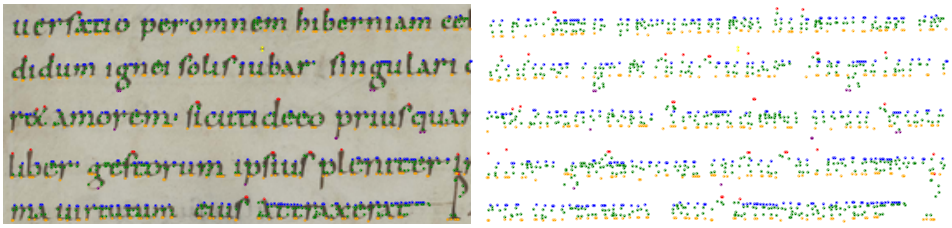


Figure 5.8: Interest Points extracted from a section of the text. The sample on the right shows the IP without the background image to illustrate the line continuity.

following stages to decide how to deal with this information[1] Unfortunately, the next IPs based algorithms work with discrete outputs.

5.4 Text Segments from Local Extrema Interest Points

Once extracted and classified the IPs of the whole page, we end up with a set of points that contains information about the location of the text reference lines. The primary objective is to join these points to extract the final text lines.

If we take a look into IPs map that is generated (Figure 5.8), the text reference lines and their directions across the IPs could be guessed. Indeed, this problem reminds to the dot-joining game for kids, where one have to join the points in a particular order to draw the final lines. However, this problem is more complicated than it appears. We will have to determine, first, a set of rules to join the points[2]. Besides, the joining approach has to deal with several issues such as adjacent text lines and misclassified points.

Since the ideal text line extraction implies to join the IPs of the same line, we followed a two-steps bottom-up approach: first, text segments by joining related IPs. Following, these groups are composed (in a second step) in order to extract text segments. Usually, we want to group IPs belonging to the same line and propagate these groups to the following text segment aggregation stage. In this step, we consider a text segment as indivisible unit taht will be

[1]The only problem comes from the data flow information between stages, since it is lighter to move pairs (`point, class`). than propagate (`point, score`$_1$`, score`$_2$`, ..., score`$_6$). But this is not a big problem because we have a relatively small set of LEPs for one page.

[2]No, a feasible solution is not to use our nephews and give them the documents to connect the dots

joined in a later step that is using higher level features. The following features are computed for each segment:

- Center point of the text segment which is computed as the average position of all IPs.
- Centerline that is calculated as the middle line between the mean line and baseline.
- Meanline (parametric form).
- Baseline (parametric form).
- Text segment area. It uses to be a surrounding polygon.

Advantages of extracting text segments from IP which track the reference lines:

- The orientation of the text segment is computed by approximating the text reference lines to knowing the orientation and the direction which will help to join the text segments according to the primary orientation of the line.
- The Noise class allows removing all the decorations and dirty artifacts within the document (Figure 5.10). In addition, if a text segment is close to (or even touching) a non-text part, the text segment will not cover the non-text areas, which it makes more valuable than other approaches.

The next problem is to find a criterion to join the IPs into text segments. On the one hand, if we are very strict when joining points (making sure they must belong to same text line), we will propagate minuscule text segments, and probably we will have problems with detecting the text orientation leading to a wrong line segmentation. On the other hand, if we are very permissive and soften the joining conditions, we will end up with segments covering several lines. We improved the text segment extraction process by applying an iterative join-and-split procedure until a reliable set of text segments is extracted.

The first segment hypothesis could be obtained by two different approaches:

- **Contours and proximity clustering** In the case of the LEP, that are extracted from contour clustering, we will use theses clusters to form the text segments.
- **Connected Components Approach** The second approach group the IPs in the same CC. The errors commited by this naive approach could

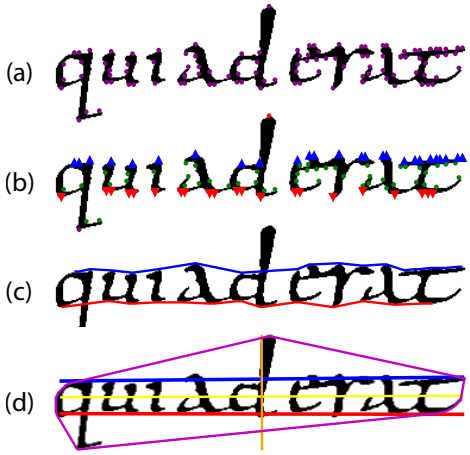


Figure 5.9: Example of bottom-up aggregation approach to text segment computation (from top to bottom): (a) Original image with local extrema detected (in purple). (b) Interest points classification (interest points belonging to the mean line in blue, to the baseline in red, and within the text in green. (c) Meanline and baseline computation from interest points. (d) Final extracted text segment.

be amended by the following text segments splitting and joining procedures. IPs are grouped by CCs. We tried to apply also a proximity algorithm like *DBSCAN* or *k-means* for clustering the IP by proximity, but we got some undesired behavior illustrated in Figure 5.11. In this case we propose to use information of the IP class to do a more clever segmentation.

Given the first set of text segments, we use the IPs and the features of the reference lines to determine whether a segment must be splitted or joined to another. For each text segment, we determine its meanline and baseline by joining the IPs associate to that reference line. Then, we apply a linear regression model, the regression score is taken as the average distance of the set of points to the regression line.

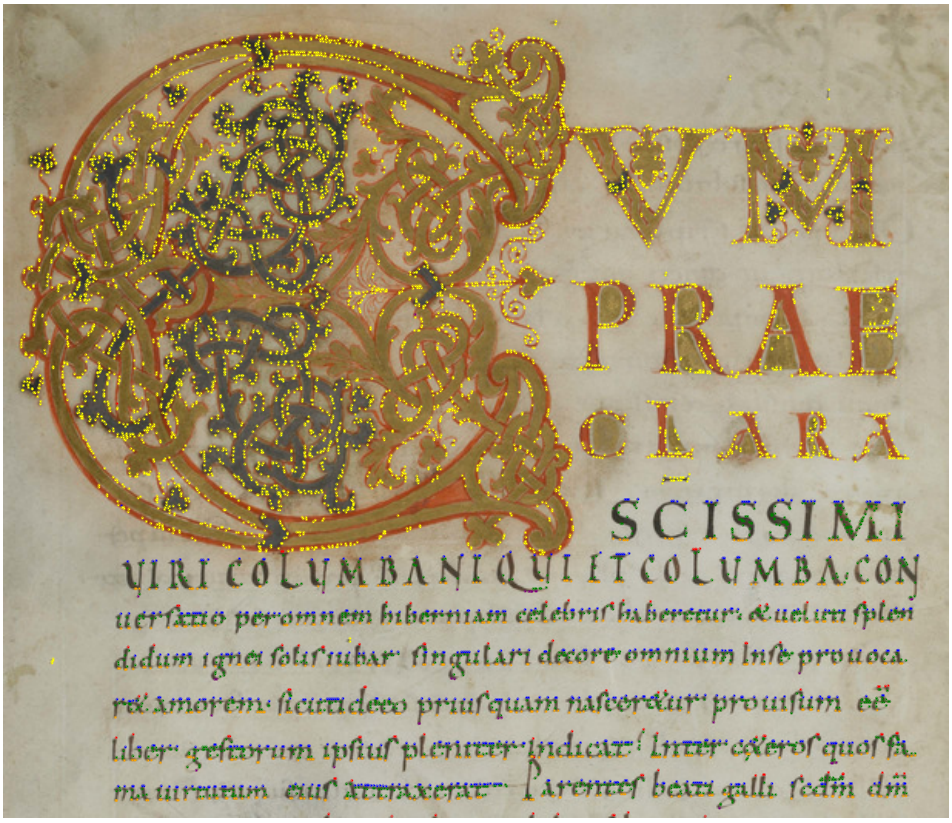


Figure 5.10: Example of the IPs classified. It can be seen than the yellow points (dirty class) are set on the decorations and not the parts of the text.

Figure 5.11: Sample of grouping IPs by proximity. In high components, the upper points are far from the lower, so they cannot be grouped by a proximity criteria. The above figure shows the IPs and each color represents the class (blue: mean line, red: ascenders, orange: baseline, green: text body). The bottom image shows the clustering of each of the points; the color now corresponds to the cluster.

Problem	Approaches
LEPs	Upper/Lower Contours Contour Clustering
IPs	Fish eye + MLP CNN
Text Segment Generation	CC clustering Contour clustering
Text Line Aggregation	Combinatorial Optimization Problem Two-by-two Segments Joiner

Table 5.3: Summary of the different techniques for Text Line Extraction based on Text Segments. 1) The Lep are extracted by Upper/Lower Contours extrema, or extrema of each contour cluster. 2) The points are classified using an MLP or CNN. 3) The text segments are build by using CCs or using the previous Contour clusters∗. 4) The final text segments aggregation.

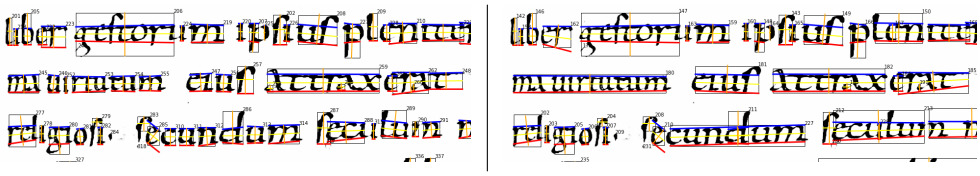


Figure 5.12: Merge procedure of close text segments The left sample shows the text segments extracted after the first clustering. The image on the right shows the segments merged according to the defined rules.

5.4.1 Merge of Text Segments

When extracting segments by CCs we still have adjacent segments corresponding to the same word, but they have not been joined because they are not in the same component. It is desirable to join these tight text segments before propagating them to the next stage. For this purpose, we took into account other features from both segments to join: proximity, the size and orientation of the components. Note that this step has nothing to do with the further text segment aggregation procedure, in this case, we want to join very close text segments to relax and improve the later computation.

In Figure 5.12, the sample on the left, shows the first estimation by extracting CCs. We have almost one component per character which can be quickly joined into words. Indeed, text segments do not necessarily correspond to one word. The merge criterion are based on the *distance, orientation, and the fitness of the IPs to the linear reference lines* (Figure 5.13):

1. First, we the candidates to merge are selected.
2. We check if one segment contained in the other. In that case, we merge them.
3. If the two segments meet the following conditions, then they must be joined:
   (a) If the MBAs of both segments intercept in the vertical axis.
   (b) We compute by linear regression the mean line and baseline for the resulting merged text segment. The fit error is the sum of the average squared distance of all the points and the line, normalized by the number of points. If the score is lower than an arbitrary

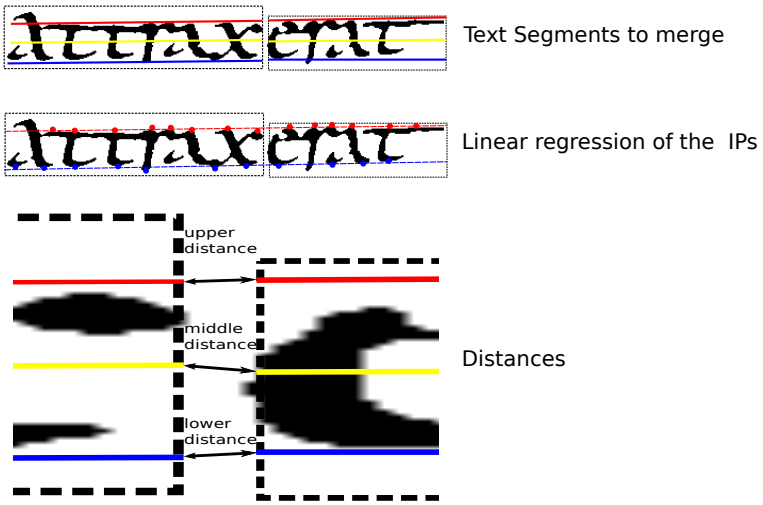


Figure 5.13: Criteria for merging Text Segments: a) fitness of the IPs to the reference lines. b) Segment distances.

threshold (between 10 and 20), we said that the two segments fit the reference lines.

$$\text{fitness(ips, line)} = \sum_{p \in \text{ ips}} \frac{\text{dist}(p, line)^2}{N} \quad . \tag{5.1}$$

(c) For the distance between segments, we compute the maximum of the three distances: upper line, lower line, and middle line distances. If the distance is less the average size of both components, they are joined. We restrict the length to a maximum value to avoid that large components "eat" the rest of elements.

Since one of the joining rules depends on the size of the components, it is common that the new segments (that were joined) could be merged again with other segments. For this matter, we apply the merging algorithm iteratively until no new segments are generated.

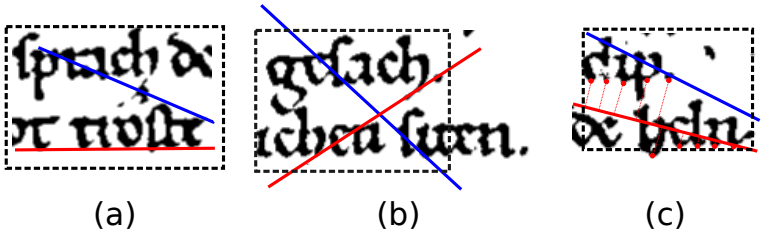


Figure 5.14: Bad extracted text segments. a) The meanline and baseline have divergent directions. b) The reference lines are crossing. c) Both lines have a low regression score regarding to the IPs.

5.4.2 Text Segments split (Touching Components)

We address now the opposite problem: a text segment does not match the fitness criteria. Using the IPs of each text segment we have an effective way to measure whether it should be splitted (Figure 5.14). Usually, the touching components issue involves ascenders and descenders from adjacent lines, or with the presence of noise (holes in the sheet, ink or bleed through). IPs are used to check the fitness of the text reference lines in the text segment. The split is computed using an iterative algorithm that is inspired by [J. Kumar, Kang, et al. 2011] and the *k-means* clustering algorithm.

First, we compute, as well, the meanline and baselines, of each CC. A segment is split if:

- When the mean line and baseline have divergent directions (< 20 degrees) (Figure 5.14-a).
- Both reference lines intercept, or the meanline is below the baseline (Figure 5.14-b).
- If the average line and baseline have a high regression score (Equation 5.1). This is usually common when there are touching components, or the segments contain text formed by several lines (Figure 5.14-c).

Then the line segment is splitted by computing new baselines. The main idea of this algorithm is to create two or more new baselines equally distributed along the y-coordinate of the text segment to split.

1. We start by re-grouping the text segments in two new segments ($n = 2$).

2. n baselines are created, they are vertical equally distributed from the highest point to the lower.

3. Baseline points are assigned to the closest of the new n baselines.

4. The new n baselines are recomputed according to the assigned points.

5. Steps 3, 4 are applied iteratively until no variation on the points assignment. The algorithm is similar to k-means since we classify the points to their closest line (like the centroids) and then, the new baseline is computed according to the assigned points (Figure 5.15).

6. The rest of the (non-baseline) IPs are assigned to each of the new segments.

    - Ascenders and meanline points are assigned to the segment whose baseline is just below them. If a point is very close to the baseline, it is not considered to be above of that.

    - Descenders IPs are assigned to the segment whose baseline is the nearest above them.

    - Text (nonreference lines points): this is similar to the meanline points, without the proximity restriction.

7. If none of the new segments fits the previous rules, the procedure is computed using $n + 1$ baselines.

8. If any of the new segments fit the rules, this is added to the list. The final segment boundaries are taken from the convex hull of the assigned IP.

We have presented an elegant solution that works quite well with most of the cases, allowing to split a text segment into 2 or more new segments, even though it is not quite common to deal with touching components that involve more than 2 lines. Nevertheless, we found one particular case where the algorithm fails miserably. This occurs when the split criteria mark a segment to be cut, but it does not correspond to several lines. This is understood better by the Figure 5.17. The problem in there is that we horizontal split have been instead of a vertical one. After splitting the segment, the 2 baselines do not correspond to the expected segments (we are not discussing if the segment must be divided or not, according to our criteria they must).

We propose a procedure to deal with this case; the first step is to detect if the split must be vertical or horizontal; this is not straightforward. One hint is that

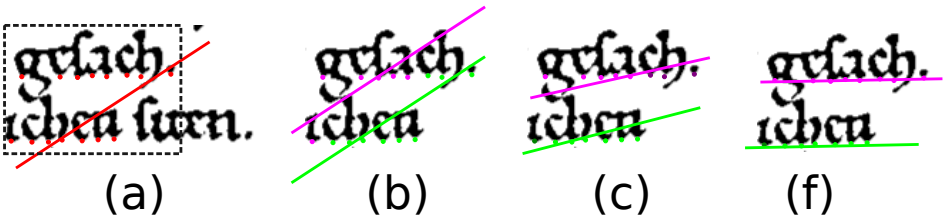


Figure 5.15: Iterative baseline computation. An Expectation/Maximization algorithm is applied like k-means to define the new baselines. a) The regression baselines computed from all the baseline points. b) The baselines are split into two lines, equally distributed. The points are assigned to the closest line (expectation). c) The lines are recomputed according to the designated points (maximization). And the points are classified again (expectation) d) The lines are recomputed (maximization) and no changes in the points classification (iterative stopping criteria).

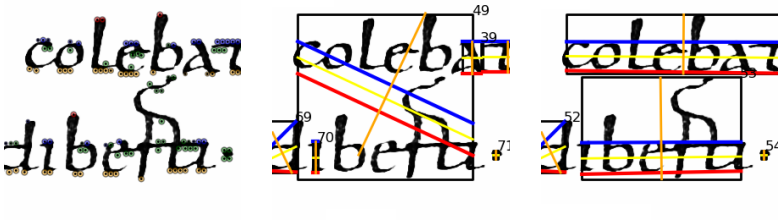


Figure 5.16: Text segment splitting. The left sample shows the Interest Points extracted on the image. The center sample, shows the merged segments, even if they are not touching component the proximity algorithm has grouped them. The image on the right shows the extracted segments after the splitting approach.

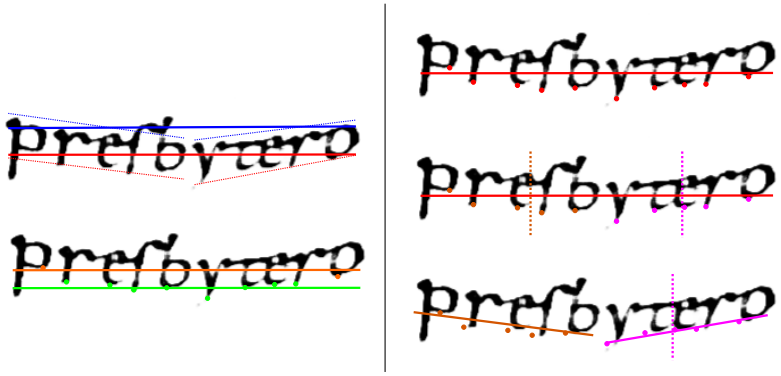


Figure 5.17: Sample where the split algorithm fails. The segment is marked to be cut. The left image shows the two baselines computed after applying the algorithm. As it can see, the assignation does not make any sense, even though the lines have a low regression score to the assigned points (brown and green). The image on the right illustrates the splitting procedure: Expectation/Maximization algorithm is applied but now, doing the points assignation to the perpendicular lines.

there are no ascender points between both baselines, but this is not sufficient evidence to detect these cases. We could apply first the vertical splitting (the one showed before) and check the regression scores of the new segments if they are higher (higher is worse) than the original one, we apply the horizontal split.

If a vertical split is detected, two perpendicular lines are created distribute them on the x-axis. Then classify the points to the closest segment, compute the new baselines, and use the new perpendicular lines for the next assignment, as depicted in Figure 5.17-right.

5.5 Text segment aggregation

When extracting text lines, bottom-up approaches usually start with text segments, and then they aggregate them by grouping based on geometric relationships among nearest neighbors; oriented nearest neighbor search; iterative grouping by proximity, similarity and direction continuity; or a combination of heuristic rules and the Voronoi diagrams. Other approaches make use of an artificial intelligence problem-solving framework using production systems, or tree structures with minimal spanning tree clustering to group CCs.

	Clean bathroom	Sweep Floors	Wash Windows
Jim	$2	$3	$3
Steve	$3	$2	$3
Alan	$3	$3	$2

Table 5.4: Toy sample of the Combinatorial Optimization Problem. There are several agents and tasks and we want to find the one-to-one assignment which minimize the total cost.

5.5.1 Combination and Optimization problem

The following section has been extracted from the collaboration work named "Combining Learned Script Points and Combinatorial Optimization for Text Line Extraction" [Pastor-Pellicer, Garz, et al. 2015] with Angelika Garz and Rolf Ingold from the DIVA group at University of Fribourg. The work combined the previous text segments, and it applied a combinatorial algorithm for the text segment aggregation problem. The procedure is shown to illustrate how the text segments could be joined, but we must credit the team at Fribourg.

Regarding a text line as a string of consecutive segments (word segments), we reformulate the problem of bottom-up text line creation as a combinatorial optimization problem, which finds one or several optimal objects from a finite set of objects. More specifically we consider the optimization as linear assignment problem. This is equivalent to finding the optimal mapping of agents to tasks, assuming that there are different costs involved for each combination [Munkres 1957]. The optimal assignment minimizes the total cost while covering all jobs and agents (1:1).

To fix ideas, let us consider a typical assignment problem using a cleaning scenario: we have three different cleaning tasks, washing the windows, sweeping the floors, and dusting the surfaces of a room. We have three workers (agents) who each demand different pay for each of the tasks. The problem now consists of finding the lowest-cost way to assign the tasks to the workers. To solve it, the problem can be represented as a matrix of agents and tasks with their respective prices, and we can find the cheapest assignment by applying the Hungarian algorithm [Y. Li et al. 2008], which solves the combinatorial optimization in polynomial time (Table 5.4).

Approaching the problem in this manner, we allow for arbitrarily curved text lines since the optimal assignment will be found given a suitable formulation of a cost. One constraint is that text lines have to follow a common pattern, i.e. intersecting or crossing lines cannot be processed; however, lines of multiple

	Segment-1	Segment-2	Segment-3	⋯
Segment-1	$\omega(1,1)$	$\omega(1,2)$	$\omega(1,3)$	⋯
Segment-2	$\omega(2,1)$	$\omega(2,2)$	$\omega(2,3)$	⋯
Segment-3	$\omega(3,1)$	$\omega(3,2)$	$\omega(3,3)$	⋯
⋯	⋯	⋯	⋯	⋯

Table 5.5: Now the tasks and agents are segments. The one-to-one assignment means that a *agent*-segment is neighbor of the *task*-segment. ω is the cost function of the assignation.

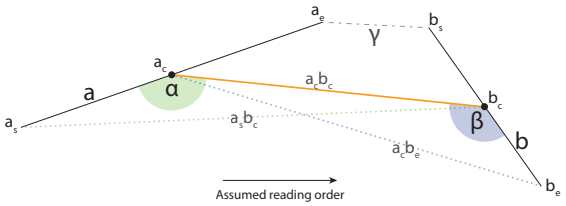


Figure 5.18: Illustration of the computation of the line continuity term δ and the distance γ. Two word segments (a,b) are illustrated by black lines that denote their primary orientation and width in this direction. Their center, start, and end are denoted by the subscripts c, s and e, respectively. The angles α and β describe the angle between the respective orientations of the segments and a hypothetical continued line between them. The shortest distance γ between two segments is illustrated as line a_e-b_s.

orientations can. The second constraint pertains the direction vector of a word segment, which has to follow the line.

As described in previous sections, we consider word segments as tasks and agents alike, defining the assigning problem as finding exactly one neighbor entity in a sequence (text line), i.e., for each entity, we find its neighbor in reading order as depicted in Table 5.5.

The reading order is determined by the median orientation of all segments on a page. Note that the actual reading order is irrelevant for the line concatenation, i.e., left-to-right and right-to-left are fungible. A segment's optimal neighbor can be itself, i.e. the end of a text line is reached, or it is noise.

Cost function For the computation of the cost function ω, three values are computed for each word segment: its main orientation o, and width and height with respect to o. We define ω as a linear combination of four factors:

1. *line continuity* δ, which captures the requirement of a text line to be smooth, i.e. no sudden changes of direction,
2. the shortest Euclidean distance γ between two segments, i.e. the shortest distance between the two closest points of the respective segments,
3. a penalty term ϵ that penalizes entities smaller than the median height of segments to ensure robustness to noise, and
4. a degree of freedom ζ to account for accuracy of the segments' orientation vectors, their position on the line, and fluctuation of a text line, which is dataset-dependent.

The cost function ω is then computed as follows:

$$\omega = \frac{(\delta * \gamma + \frac{\gamma}{\rho}) * \epsilon}{\zeta} \tag{5.2}$$

with ρ being dataset-dependent and regulating the influence of distance γ with respect to the line continuity term $\delta = \min(\alpha, \beta)$, where α is the angle formed by the lines $a_s a_c$ and $a_c b_c$, and β being defined analogously, and ϵ is defined as

$$\epsilon = \begin{cases} 1, & \text{if } h_w > h_m; \\ 1 + \frac{abs(h_w - h_m)}{(h_m * \lambda)}, & \text{otherwise,} \end{cases} \tag{5.3}$$

where h_w and h_m are the heights of the word segment and the median segment height, respectively, and λ regulates the total influence of the penalty term. Figure 5.18 illustrates the calculation of the line continuity term δ and the distance γ.

Post-processing Having joined the segments to lines, we use unsupervised recursive k-means clustering as a post-processing step to reject invalid text lines. The underlying assumption is that text lines are organized in blocks of similar appearance. We use normalized features such as the length and height of a text line's bounding box, its ratio, the number of word segments, and the overall orientation of the text line.

First, we use $k = 1$ and determine the best cluster center out of several attempts by homogeneity of its cluster, which is defined as the smallest average distance ($\mathrm{mean}(t_d)$) of all text lines t to the cluster center. Should the cluster be too inhomogeneous, i.e. $\mathrm{median}(t_d) > 0.85 \wedge Q_3(t_d) > 1$, we increase the number of clusters (Q_3 is the middle value between the median and the highest value of the data set). Having selected the best cluster center, we recursively apply following two steps n_{rec} times: first, removing those text lines where $t_d > th$, with th being a threshold that starts at 1.2 and is lowered with each recursion by 0.1, and second, clustering the remaining lines with k-means. Thus, we keep a set of homogeneous lines and reject outliers. Depending on the reliability of the input data (segments), the parameters of the aggregation method need to be adapted. The degree of freedom ζ and ρ are determined on a single page of the dataset.

Baseline: Connected Components Text Segments The Combinatorial Optimization approach have been tested with the text segments extracted by tracking the text reference lines (IPs). IPs allowed us to extract reliable text segments, since they check their integrity by the joining/merging algorithm and discard non-text components as well. And, what is more important, we computed a proper orientation of the text segments by extracting the reference lines of each text segment.

At this point, a question comes to our mind: *Do we need all this computation of* LEP*s and IPs? May we do better or equal using a much simpler nonsupervised technique?*. For this purpose, we tried to extract directly text segments from CCs. Each CC is considered one text segment, and calculating the text boundaries is straightforward: use the convex hull (or smarter concave-hull) of the component; the problem appears when trying to calculate the orientation.

For computing the direction, we followed a similar idea from [Ouwayed et al. 2010]: Vertical and horizontal histograms are calculated for each text segment. The histograms are computed at different angle ranges: $[-45°, 45°]$. We can see the output of the histograms as a 2D matrix (angle range $\times$ histogram range). The optimal angle is computed applying the *Wigner-Ville* distribution

function over the mentioned matrix. Usually, with the horizontal profile, the orientation works well for most of the text segments, but we added the vertical one for segments like “I” or “l”.

Figure 5.19 shows the histogram procedure for a text segment. The *Wigner-Ville* gives one score for each of the possible angles, and then the maximum value is the detected text orientation. It is worth to mention that the larger the text segments, the more accurate the orientations are.

5.5.2 Two by two segments joiner

This section has been the result of the collaboration of Jérémie Bosom from Université Paris-Sud in our research group. The work presented here was part of his internship during the 5th year of his engineering courses.

In this section, we explore an alternative approach for the text segment aggregation problem. We will try to join the detected segments by pairs. For that, we propose two approaches: one using a heuristic or set of rules to join the segments and the second that uses a ANN for this purpose.

Most of the text segments aggregation approaches rely on finding the right neighbor of each component. They start building the text line aggregating text segments from left to right. Local and global information are combined in order to find the next neighbor by comparing the height, distances, and orientation of the text segments respect the averages.

But now, the text segments extracted contain more useful information: size, orientation, base/upper/center-line, etc. Our goal is to analyze if it is possible to use a two-by-two segment joiner according to our previously extracted text segments. Joining pairs of text segments without global information is a dangerous assumption. Therefore we will have to invest in post-processing stages to fix possible errors during the joining. For a text segment the potential neighbor candidates are evaluated. Then for a pair of text segments a set of features from the two segments is extracted in order to decide if they correspon to consecutive text segments of the same line.

The classifier, in principle, is very simple: it receives 2 segments (or features regarding both segments), and the output predicts if the second segment is the right neighbor of the first. This approach adds some challenges we have to deal with:

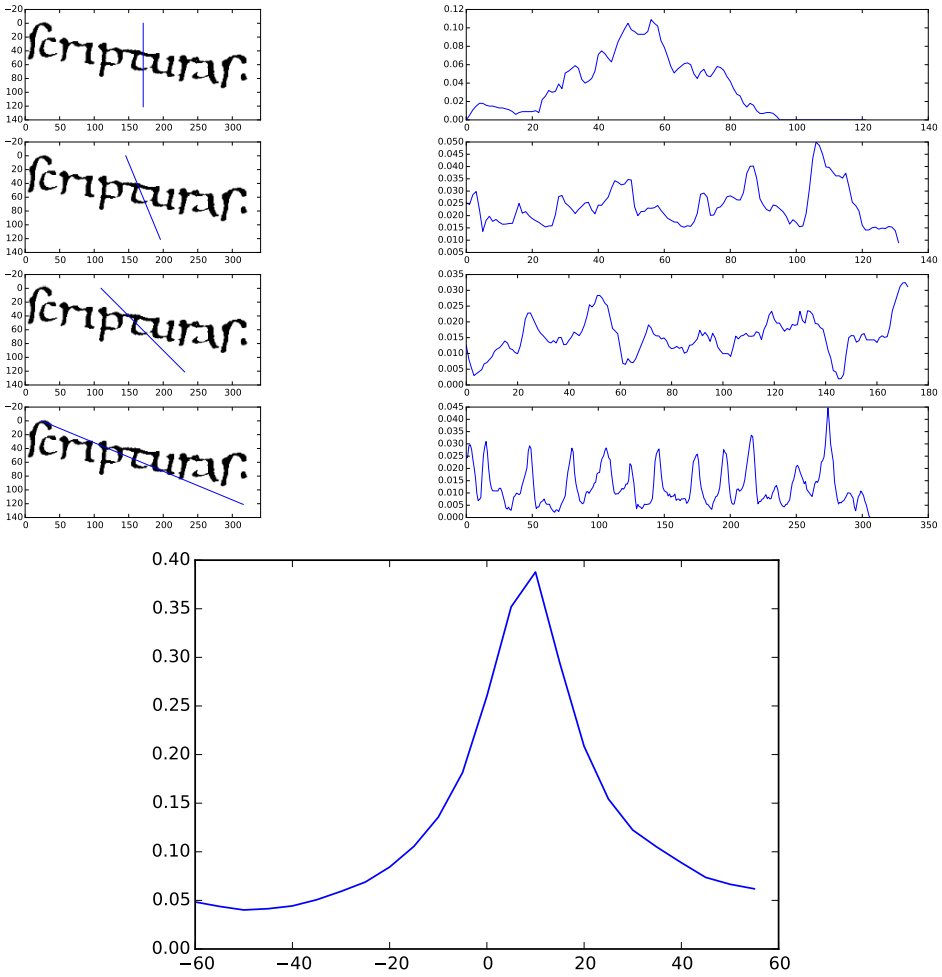


Figure 5.19: Top graphs show the histogram profile of the CCs for several angles. The bottom shows the Wigner-Ville distribution function for the previous profile. Checking the function we can see a maximum value at 10 degrees.

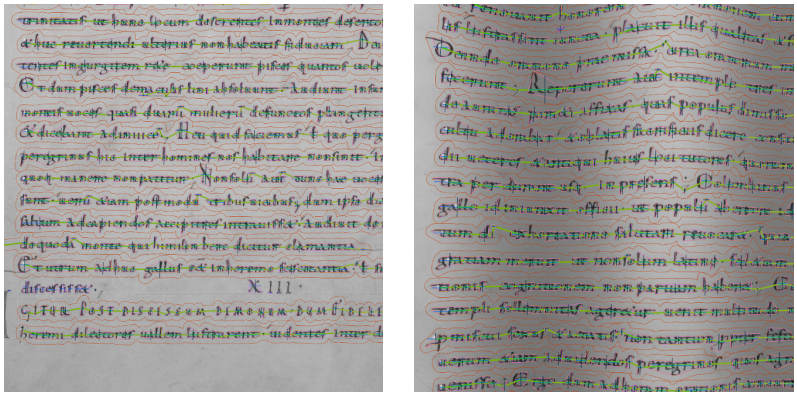


Figure 5.20: The left sample corresponds to an image of the Saint Gall Database. The right image shows the lines extracted for a sample in the distorted version. The green lines illustrate the connected segments joined for computing the lines (Extracted from [Pastor-Pellicer, Garz, et al. 2015]).

- *Which features should be used by the classifier?* Instead of using text segment features, we need to use data about the interaction between both: distances, orientation, position, size, etc.
- *Check the candidates of each segment.* We have to decide the criteria for extracting pairs of potential neighboring segments.
- *Create the text segment joiner ground truth.* Once we know the candidates and the features, we need supervised data for training the classifier. We should provide positive and negative samples to the classifier.

Also, we have to control 2 assumptions:

- A text segment has at most one *right* neighbor.
- A text segment has at most one *left* neighbor.

In case we have more than one neighbor, the output of the ANN is taken as a confidence value.

We propose first, a joiner that uses a set of rules based on the features extracted from the pairs of text segments. And second, we use a neural classifier that computes if two segments must be joined.

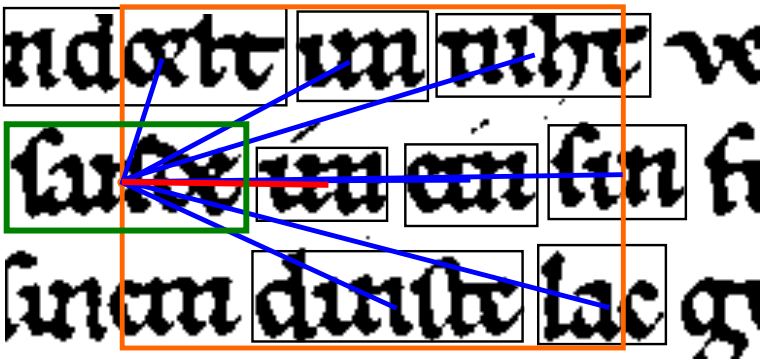


Figure 5.21: Candidates taken for the two-by-two Joiner. In green the segment to be joined, the orange box shows the candidate searching area. Blue connections indicates not right neighboring and the red one the correct neigbor.

Candidates function Before deciding which features are useful for the segments joiner, it is important to define how to extract the segments candidates to be merged. We need a reliable function, which given one neighbor, it can remove potential candidates to be combined. Besides, we should avoid irrelevant candidates to save useless calculations of features.

Our first solution to find candidates is to consider all the segments within all the square/circle area around the candidate text segment. This solution is not useful for three reasons:

- The function is reciprocal. So we will have duplicated candidatures.
- Some of the components in the area are irrelevant, and it is straightforward to discard them.
- Increasing the area to find more suitable candidates leads to add more irrelevant candidates than relevant.

For the candidates searching function, we used a rectangle. The rectangle is settled next to the current text segment, and the box size is proportional to the segments. To determine if a segment is within the searching box, it must have its middle point inside the rectangle. Figure 5.21 illustrates the searching area, all components within the orange box are candidates to merge with the left segments.

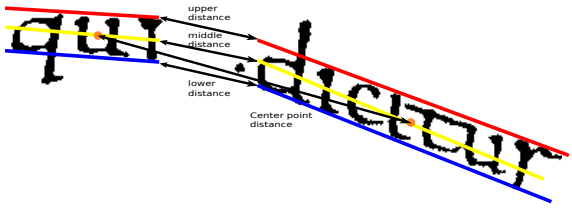


Figure 5.22: Distance features computed for a set of segments.

Extracted features Once the two text segments are selected, it is important to choose which features are relevant to decide whether both segments must be merged. According to the text segments extracted on Section 5.4, we could select the features from the pair to be joined according to distances and orientation between both text segments; the MBA of the segments and its direction. Regarding to to the distances between segments, the following features are used (Figure 5.22):

- **Base distance (d_{base})** The minimum distance betweem baseline and meanline.
- **Top distance (d_{top})** The minimum distance between the two meanlines.
- **Center distance (d_{center})** The minimum distance between the center lines.
- **Average Points distance ($d_{avgPoint}$)** Distance between both center points.

We add more features related to the sizes of the component and their MBAs.

- **Area of the text segment** The size of the bounding boxes of the segments are added as features.
- **Area of the MBA ($pMBA$)** Two more features (one per segment) are included with the size of the MBA. This area corresponds to a trapeze. (Figure 5.23).
- **Projection of the MBA** The first MBA is projected into the second as seen in Figure 5.23. We force the meanline and baseline of each segment to have the same orientation. That is to change the slope of the meanline to the baseline.

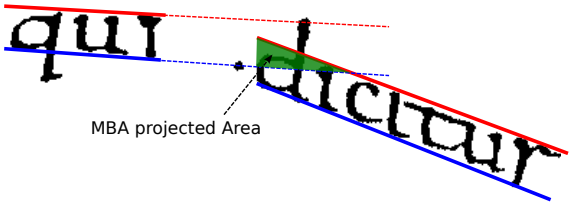


Figure 5.23: Main Body Area projection of one segment to the joining candidate. The features is normalized by the total MBA of the candidate.

Heuristic Joiner If we check the segments forming a line, we could observe that most of the time the right segment is the next neighbor in the line. Another assumption is to expect some homogeneity on the same page (or even corpora).

For this purpose, our Heuristic classifier will use a Gaussian distribution function between the features among the closest text segments on the page. When having two segments to classify we compute the Gaussian functions of each of the features according to the average (μ) and its standard deviation ($sigma$) from the whole page. The function is split in two sub-functions: If the bounding box of the text segments overlap, we apply the joining function as:

$$P(join(s1, s2)) = G(d^{s1,s2}_{avgPoint}, \mu_{d_{avgPoint}}, \sigma_{d_{avgPoint}}) \tag{5.4}$$

If not we compute the score function with the rest of features:

$$\begin{aligned} P(join(s1, s2)) &= \phi_{d_{center}}\, G(d^{s1,s2}_{center}, \mu_{d_{center}}, \sigma_{d_{center}}) \\ &+ \phi_{d_{base}}\, G(d^{s1,s2}_{base}, \mu_{d_{base}}, \sigma_{d_{base}}) \\ &+ \phi_{d_{top}}\, G(d^{s1,s2}_{top}, \mu_{d_{top}}, \sigma_{d_{top}}) \\ &+ \phi_{pMBA}\, G(pMBA^{s1,s2}, \mu_{pMBA}, \sigma_{pMBA}) \\ &+ \phi_{d_{avgPoint}}\, G(d^{s1,s2}_{avgPoint}, \mu_{d_{avgPoint}}, \sigma_{d_{avgPoint}}) \end{aligned} \tag{5.5}$$

The total score is taken by a linear combination of the different Gaussians functions. Nevertheless, $\phi_{d_{center}}$, $\phi_{d_{base}}$, $\phi_{d_{top}}$, $\phi_{ProjMBA}$ and $\phi_{d_{avgPoint}}$ can be tuned for better results. In the typical case, they have the same weight ($\frac{1}{5}$).

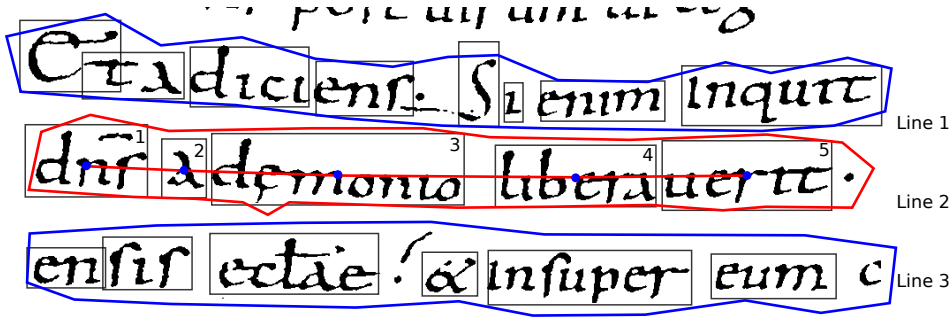


Figure 5.24: Sample of how the ground truth of the segment joiner is generated. First, each segment is assigned to a line; then they are sorted by the x coordinate. Finally, each segment is marked as right neighbor of the previous text segment.

Artificial Neural Network Joiner The second approach consists of replacing the heuristic classifier by a ML approach. Following this PhD Thesis line, we opted for the use of ANNs.

The set up is very simple, the input will be the features described above, and the output corresponds to a binary output where 0 means the two segments are not joined, and 1 to join them. The model output is a logistic neuron, where we have as output the estimation whether the two segments must be joined.

Generating the ground truth The first problem of this approach relies on the generation of a suitable ground truth for training the supervised model.

The input of the classifier are text segments computed by joining IPs extracted from the text. These segments are part of our approach, but they are not present in ground truth of the lines. In other words, we do not have a ground truth saying what a text segment is, neither their position on the text line and its right neighbor.

Instead, first, we assign each text segment to one of the text lines. In the case that a text segment overlaps two lines, we add it to the line whose overlapping area is higher. Then we sort all the segments of a line by the x coordinate (average point) to define the right neighboring as depicted in Figure 5.24. Thus these pairs of text segments (one segment and its right neighbor) are added to the ground truth as positive samples. Regarding the negative cases, we use the rest of text segments according to the candidate's function utilized in the heuristic joiner. Each candidate segment which is not the right neighbor is added to the ground truth as a negative sample.

Post processing Finally, for both approaches, we could apply a basic post processing for remove some false positives regarding:

- In some borders of the page there have been classified some IPs as text. There are some small dirty text segments marked as text where are not part of any line.
- Small components, like accents, which had not been merged.

For this purpose all the lines with a width lower than a certain threshold (usually 500 pixels) are skipped. We found some other issues:

- Tiny components are difficult to merge.
- Some lines are identified as two or more lines instead of one.
- There are a lot of false positives.
- Some text segments between lines, like accents, tend to stay alone or add them to the wrong line.

Some specific treatment should be used to fix these issues:

- The first consists of identifying parts of the text that have been detached from the lines. For this purpose, we rely back on the IP. We check segments that contain descenders but not ascenders or meanline IPs. If this is the case, we merge the component with the closest element above it. In the same way, we join segments containing ascenders but not descenders nor baseline points with the nearest element below it.
- Some lines have been split (or not merged) because they did not follow the merging criteria. It is easy to check if very close lines correspond to the same line:
  - If the lines cross each other.
  - If the distance between lines is smaller than the average distance of all components of the page.

Summarising, first, merge small components to the closest line. Second, we merge medium text segments with bigger lines. And finally, large components corresponding to the same text line are merged.

5.6 Text Line Extraction from MBA estimation

In this section we introduce an alternative approach for TLE. Instead of tracking the text reference lines, now we detect the Main Body Area (MBA) and the lines are extracted from it. In this approach, CNNs, which have learned useful features from the binarized-free document images, are used for extracting text lines, by detecting the MBA. This approach works better when it is applied to text blocks (usually columns) that are extracted in a previous layout analysis stage. Each text block is independently tackled and the CNN scans it generating an MBA map. From that map, a segmentation algorithm is applied in order to extract the lines. Finally, a final post-processing to enhance the frontiers from the lines is applied.

The main contribution of this work is the practical use of CNNs directly on images of historical documents and its robustness to noisy inputs and other problems such as touching components. The proposed method estimates the MBA of the text lines at page level. The underlying idea is to classify each pixel from the raw image within the probability of being part of the MBA. The MBA follows a continuous left to right path which provides a good estimation of the position of the lines and their orientation, even if the page is skewed or the typography presents slant. Also, the MBA does not cross along different lines, which makes it robust against touching components where some traditional methods fail.

Indeed, CNNs can extract this shape features giving high performance results on the MBA classification as the shape of the letters within the MBA is regular within different characters (i.e., letters “o”, “b”, “d”, “g”, “p”, “q” have a very similar shape in the MBA, they all are like “o”), the MBA computation is very suitable for working with ML techniques.

This approach is inspired by the work introduced in [Baechler, Liwicki, et al. 2013] since the lines are extracted after classifying pixels on the text core line class. Nevertheless, the differences are substantial, Baechler et al. used two classification levels: first, pixels are classified as decoration, background, text block or periphery classes; and then, with a higher resolution input, the outcome of this first stage is used for a more fine classification into decoration, background, and core-text line. Following the same procedure, we detect the text blocks in the first stage, and then we perform the MBA extraction only in the zones marked as text blocks. The main novelty of our approach is the use of CNNs over the raw input of the image instead of feature based models. Also, our method works well using only 2×2 downsampled images. Moreover, the

line segmentation from the text core line also differs from the original work, since the output of the CNN presents some continuity that does not require further components joining nor the post-processing used in Baechler et al. work. Besides, our ground truth has been generated by means of bootstrapping techniques and transferred knowledge combined with semi-supervised methods (as detailed in Appendix E).

5.6.1 Text Block Extraction

As mentioned, the proposed method works in two classification stages. On the first stage, every pixel of the image is classified into one of three classes: text block, background or decorations/graphs. Following, a second CNN is applied. Even though our approach could skip the first layout classification, it provides several advantages:

- Without the first classification, the MBA classifier would have to deal with less specific data, since it removes decorations and many not useful data like background pixels. The model gets specialized into discriminate text parts from the body area and the rest of zones.
- In this way, the MBA classifier runs only in the text blocks areas, which usually takes between 20% and 80% of the whole page. Thus, deeper and more robust models are used in this second stage.

As for drawbacks, the Layout or text block ground truth is needed to extract the text blocks, and it is required to perform a full page scan for this first classification.

Figure 5.25 illustrates the whole text block pipeline from the original image (Figure 5.25-a). After running the Layout CNN all over the page, the pixels are labeled as text block if its CNN output value is greater than certain threshold set to 0.5 (as shown in Figure 5.25-b). Then a text block map is generated (Figure 5.25-c), and the text blocks are delimited by joining neighboring pixels that are marked as text (8-connectivity). Usually, this procedure segments text columns, but also, small and spurious text regions must be filtered according to their size. Finally, a gap filling algorithm is performed on the text blocks to have a closed polygon shape (Figure 5.25-d). The text line extraction is carried out independently in each of these text blocks (see Figure 5.25-e).

(a) (b) (c) (d) (e)

Figure 5.25: Text block pipeline. a) Original image. b) The softmax output after applying the Layout CNN. c) Text block pixels after applying the threshold. d) Joined text blocks with the gaps filled. e) Main Body Area map computed over the text blocks.

	Layout CNNs	MBA CNNs
Input window	37×37	43×43
Convolutions	10 $(6 \times 6 + 1 + 1)$ kernels 2×2 max pool (ReLU) 20 $(4 \times 4 + 1 + 1)$ kernels 2×2 max pool (ReLU)	10 $(6 \times 6 + 1 + 1)$ kernels 2×2 max pool (ReLU) 20 $(4 \times 4 + 1 + 1)$ kernels 2×2 max pool (ReLU)
MLP	128×16 ReLU	128×16 ReLU
Output	3 softmax	1 logistic

Table 5.6: Convolutional Neural Networks Configuration for MBA and Text Block.

5.6.2 Main Body Area pixel classification

The key point of the proposed approach is to classify the pixels of the image belonging to the MBA. Performing the MBA classification follows a similar procedure than the one in the layout classification level. The main difference is that now we have fewer pixels to classify, and the output of classifier has only one logistic neuron as output. Table 5.6 describes the CNN configurations for both stages.

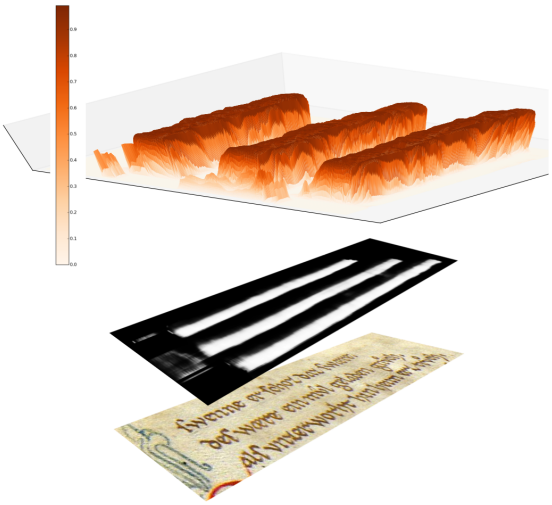


Figure 5.26: The watershed transform tackles the MBA map as a set of valleys and mountains. Then it tries to find the local minima height level flooding the map; the high probability "mountains" are the final segmented lines.

5.6.3 Text Line Segmentation by Watershed Transformation

After estimating the MBA map for the whole page, we can observe in the samples (Figure 5.27-a) that the text lines have been mainly detected. Hence it would be easy to extract the lines from it.

We tackled this problem as an image object segmentation, where the objects to segment are the lines in the MBA. Following this reasoning, a region-based segmentation algorithm using the watershed transform is applied [L. Vincent et al. 1991]. Roughly, the watershed treats the MBA probability map as a set of valleys and mountains according to the output values of the CNNs. It tries to find a local minima level flooding these valleys and preserving the mountains which are zones of high MBA probability, which are the final objects segmented lines (Figure 5.26). An alternative naïve approach is to use CCs, even though this idea requires thresholding techniques and it does not explode the soft information propagated from the logistic neurons.

Figure 5.27 illustrate the procedure followed for one text block:

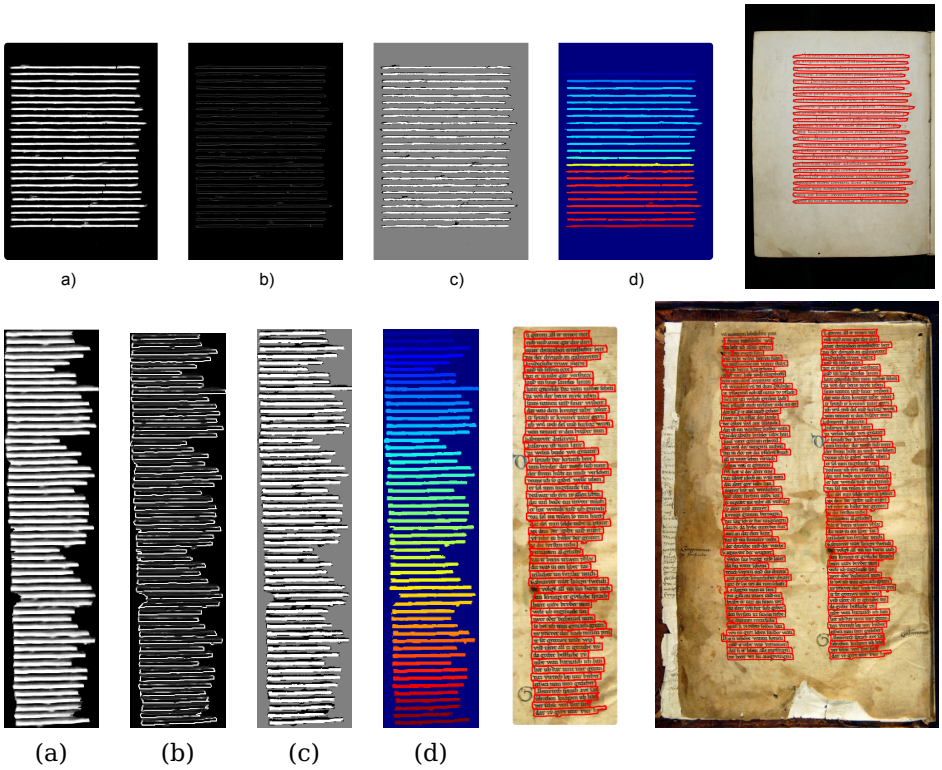


Figure 5.27: Text line segmentation process for one of the text blocks from Figure 5.25. a) MBA map computed by the CNN. Brighter values represent the pixels belonging to the MBA. b) A Sobel filter is applied for computing the frontiers of the lines. c) Markers are computed classifying pixels from (a) in three classes: values less than 0.2 (gray), between 0.2 and 0.7 (black pixels) and above (white pixels). d) The regions centered on the MBA of the lines. The frontiers of the lines after post processing the MBA regions and the lines extracted from two text blocks (columns in this case) are shown in the last sample.

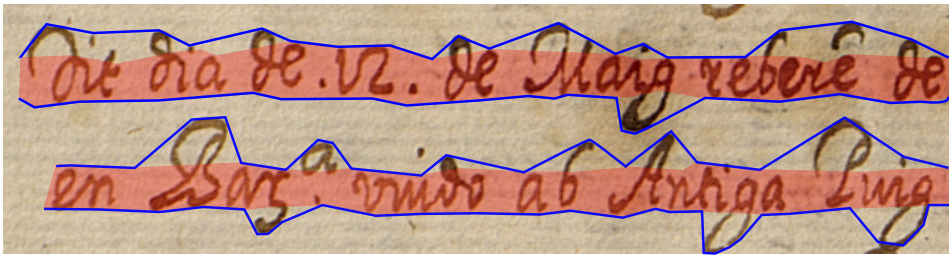


Figure 5.28: The new frontiers of the lines are enhanced by detecting and adding the correspondingLEP which are located outside of the MBA. Red areas delimit the shapes found by the watershed segmentation algorithm. In blue, the new limits when addind the new points to the original line.

1. First, a Sobel filter is applied (Figure 5.27-b) on the MBA map (Figure 5.27-a) in order to compute an elevation profile.

2. Since the values are probabilities computed by the CNN, the MBA is separated into three classes (markers): values less than 0.2; values between 0.2 and 0.7; and values above to 0.7 (Figure 5.27-c). The watershed procedure needs these markers.

3. The markers are used by the elevation map to apply the final watershed transform. The watershed transform detects several regions which correspond to a line (Figure 5.27-d).

5.6.4 Post-processing

The segmentation algorithm extracts tight polygons surrounding the MBA of each of the detected text lines. Even though the algorithm can identify most of the text lines, we need to provide more accurate frontiers, taking into account ascender and descenders. For this purpose, a simple procedure is applied: on the text blocks, the LEPs are extracted and added to the line polygons. Some of these LEP delimit the frontiers of the text lines, the points which are not inside of any polygon area are added to the line frontier if the following conditions are met: the point must be local maxima and the distance to the next polygon (which is below to it) should be lower than a certain threshold (usually 10 pixels). The same procedure is applied to the minimum extrema points which are joined (or not) to their top closest lines. Figure 5.28 illustrates this procedure.

5.7 Experimental setup and results

Finally, we have tested the presented approaches with the IAM Historical Document Database described in Appendix C.2. The evaluation metrics used in here are detailed in Appendix D.1.

With the following experimentation we aim at the following objectives:

1. To verify and evaluate the proposed methods with the aimed corpora.
2. To compare the supervised approaches with equivalent heuristics.
3. To compare the performance of our models with other techniques in the literature.

We grouped the methodologies developed in three main categories:

- **Combination and Optimization Problem (COP)** (Section 5.5.1) This is the approach that computes text segments and the combine them to create the final lines by optimizing a joining function.
  - *Proposed*: the proposed approach that uses IPs for extracting the segments.
  - *CCs-based*: using CC as text segments as a baseline comparison.
- **Two by two segments (2-by-2)** (Section 5.5.2) Here the segments are joined using neighbor classifier.
  - ANN: It uses an MLP classifier instead.
  - *Heuristic*: Uses a rule based approach (baseline).
- **MBA estimation (MBA/CNN)** (Section 5.6) extracts the MBA of the lines and segment them by the watersheed transformation.[3]
  - Proposed: Uses a CNN for extracting the MBA.
  - RLSA-based: applies the smearing procedure on the pixels of the text blocks, and then the same text line segmentation in our approach is computed over it. Note that the RLSA approach works on the binarized ground truth of the image which has not been used in our methods.

[3]The MBA approach uses a previous layout extraction approach that computes the Text Blocks.

Tables 5.7 and 5.8 summarize the results of the proposed approaches (on the test sets). Note that it has not been possible to try all the approaches with all the available corpora. Nevertheless, it is feasible to observe the performance of each approach versus their corresponding baseline alternative.

The COP approach results have been shown in another table (Table 5.8) since they have been evaluated according the TLL-DR/PHR based metric.

			Match Score metrics			Pixel Accuracy		
Dataset			DR	DA	FM	Prec.	Rec.	FM
Parzival	MBA/CNN	Proposed	98.65	98.86	98.75	98.61	98.90	98.79
		RLSA-based	50.00	70.22	58.45	72.62	68.41	70.45
	2-by-2	ANN	80.39	79.27	79.83	91.82	90.26	91.03
		Heuristic	91.91	91.03	91.44	95.98	95.04	98.79
Saintgall	MBA/CNN	Proposed	96.39	96.52	96.46	99.06	98.49	98.76
		RLSA-based	89.59	92.55	91.05	94.40	95.65	95.01
	2-by-2	ANN	93.75	76.19	84.07	96.9	95.69	96.01
		Heuristic	94.84	81.24	87.33	96.36	96.16	96.18

Table 5.7: The table compiles the performance of the proposed methods on the different test sets. The parameters selected for better understanding of the performance have been the Match Score metrics and the Pixel Precision and Recall.

Dataset	COP models	TL-PA	TLA
Saintgall	Proposed	98.3	97.2
	CCs-based	97.6	89.5
	Geometric IPs	98.0	97.2
DL Saintgall	Proposed	98.0	96.7
	CCs-based	96.9	88.3
	Geometric IPs	97.8	95.9
	[Garz, Fischer, Bunke, et al. 2013]	97.44	93.99

Table 5.8: PA and TLA for the models used in the Combination Optimization Problem approaches.

Furthermore, Table 5.9 comprises the performance of our approaches compared with other reported works. However, getting a fair and transparent comparison has been an impossible quest since different authors used different sets, ground truths, different metrics, even different implementations of the assessment toolkit could lead to differences. So the results shown here are merely indicative.

Corpora	Method	PHR	tll-dr
Parzival	[Baechler, Liwicki, et al. 2013]	96.3	96.4
	[Cohen et al. 2014]	98.31	99.22
	MBA/CNN	98.79	98.76
Saint Gall	[Baechler, Liwicki, et al. 2013]	96.0	96.4
	[Diem et al. 2013]	98.94	99.03
	[Rabaev et al. 2013]	-	97.84
	[Garz, Fischer, Sablatnig, et al. 2012]	98.65	97.97
	[Cohen et al. 2014]	99.08	99.22
	MBA/CNN	98.49	96.4

Table 5.9: Comparison with other works. Pixel Hit Rate and Text Line Accuracy are the selected metrics for comparison.

5.8 Layout Analysis

The content of the following section was developed during my internship at the University of Kaiserslautern by the guidance and advice of Marcus Liwiki and Shan Afzal.

The proposed method in Section 5.6 labels the image pixels as text, decoration/graphics and background in the first step. Somewhat it corresponds to a Layout Analysis stage.

We have tested the potential of our one-to-one labeling method (Section 3.7) in the context of layout analysis by means of labeling parts of the document images, i.e. text, non text, decorations, graphs, etc. The convolutional configuration used to this task is described in Table 5.6, and like other high/medium resolution images a trade-off between optimal convolution topology and efficiency plus performance is required. As seen, these configurations gave us good outcome for further text line segmentation.

The output of the classifier has 3 softmax output neurons, each of them corresponds to the class: background, decoration or text block. The final evaluation is performed by comparing the labeled pixels between the input and ground truth. Pixels that are not part of the page have been merged into the background class. If the image has enough resolution, we reduce the image by a factor of 2×2, although the final evaluation is always performed on the original scale.

For this purpose, we the Saint Gall and Parzival datasets; and, the UW-III dataset which consists on current technical images. The following tables com-

pile the results on the proposed corpora. MDLSTM[4] have been used following the same approach showed in Chapter 4.5, then we combined the two approaches like in the binarization task

Class accuracy (All pixels)

		Text as Text	Graph as graph	Bg as bg	Accuracy
Saint Gall	CNN	98.37	76.89	98.46	91.36
Parzival	CNN	94.23	67.70	99.63	87.19
	MDLSTM	96.29	48.75	98.11	81.04
	Avg	92.68	32.04	99.77	74.83
	Best	95.37	56	99.44	83.6
UW III	CNN	97.86	79.66	97.86	91.79
	MDLSTM	97.34	73.82	97.34	89.50
	Avg	98.86	76.52	98.86	91.41
	Best	97.996	77.61	97.99	91.2

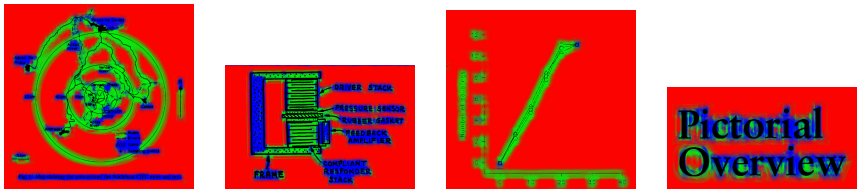
Class accuracy (Only foreground pixels)

		Text as Text	Graph as graph	Bg as bg	Accuracy
Saint Gall	CNN	99.71	80.9	23.04	67.87
PARZIVAL	CNN	99.89	82.49	96.7	93.03
	MDLSTM	99.72	56.534	93.44	83.23
	Avg	99.86	45.42	97.43	89.68
	Best	99.96	72.42	96.64	66.13
UW III	CNN	98.60	94.28	-	96.44
	MDLSTM	95.72	93.58	-	94.65
	Avg	99.43	91.43	-	95.43
	Best	98.20	94.70	-	96.45

Table 5.10: Overall Accuracies for the pixel labeling layout evaluation.

Table 5.10 measures the accuracy for each class, the overall efficiency is computed as their averaging. But, the results must be reviewed carefully, for example, the Background accuracies are usually high since is that the major part of the pixels that are white (or not ink) belong to the background. But also there are other white pixels that are part of the graphs, as seen in Figure 5.29. A chart, for example, where the CNN input receptive field does not get enough contextual information, will be marked as background. Another interesting evaluation is only to take into account the foreground pixels as shown in the table below. Therefore e have shown the results only for the foreground pixels in the second table.

[4]Shan Afzal must be credit for the training of the recurrent models. We used them for comparison with the convolutional models and the ensembles.



a b c d

Figure 5.29: Some examples of issues that can be found in our layout analysis by applying local methods (CNN). a) Text within graphics. b) Graphic parts classified as text. c) Charts, where the inner pixels are classified as background. d) Capital and headers.

The Background recall is mis-leading, since a background pixel cannot be part of the foreground data, in this case, we refer to stains and dirty parts of the document that are nor text neither decorations. The fraction of this issue is almost negligible which explain large variations in this value (for example for Saint Gall database). The UW-III database, instead, has been binarized, so we skip these values.

We should remark some issues found with the UW-III dataset (illustrated in Figure 5.29):

- Text within graphics: it is not clear if they should be labeled as graphic or as atext. Since our approach uses local inforamation, it tends to mark them as text.
- False positives when graphics have shapes similar to text.
- Lack of context, specifically in figures that contain big white areas.
- NonText Scale invariant. The problem in titles and small captions due to the lack of information on the training dataset.

Note that we could not compare our performance with other approaches since it has been difficult to use the same evaluation scenario. We address the reader to check the works of [Bukhari, Al Azawi, et al. 2010; Chen et al. 2015] for similar evaluation setup. In any case, this results are very interesting and show how the methods presented in this chapter could be applied to other related tasks.

5.9 Discussion

The tables presented in the previous section show a large range of measures and methods. In this section, we will discuss the system performances result and draw several conclusions.

Combination and Optimization Problem As shown in Table 5.7, the COP approaches have been tested on the distorted and nondistorted Saint Gall datasets. The majority of the errors at line level are poorly assigned lines that not pass the matching threshold (these represent the $60-80\%$ of the mistakes). Missed lines take around $10-25\%$ of the errors. These lines are primarily last lines of a paragraph which are significantly shorter than the rest of the lines. However, these errors barely affect the pixel hit rate since only a few pixels are missing. For further processing, lost lines are the most severe errors, while insertions can quickly be discarded in following recognition steps. Some text segments are not correctly joined since they are too small and they are not similar enough to the rest of the text line, so they are discarded while joining the lines. Note that no prior filtering on the connected components approach has been applied, leading to noisy data; however, competitive results can still be achieved.

This approach compresses different techniques and it has a long sequential pipeline: contours extraction, LEPs detection, IPs classification by CNNs, segment computation and finally, the text line Aggregation. The algorithm takes around 1 minute for processing one page. A normal page contains around 10000 Interest Points. It takes less than 10 seconds to classify them using the April-ANN toolkit with the *Intel MKL* library on an *Intel i5 (4th generation)* processor at 3.2 GHz with 8GB of RAM. It is worth to mention that the contours clustering process takes half of the computation time.

Two by two segment joiner As expected the two-by-two approaches showed poor performance, even with ML methods which use local information. We tried to propagate the local information given by the classifier to a global post-processing steps, but it has been not enough to reach the desired result.
Indeed, the Heuristic Joiner, in overall, provided better performance than the ANN classifier. However, the difference with the Saint Gall is minuscule at the pixel level.

Text Line Extraction by MBA Even if the text block extraction stage is not perfect, the MBA detection could recover some mistakes if the recall of the text block extraction is high. The text block step helps mainly to speed up the MBA classification and also for segmenting columns, which is the case of Parzival dataset. The recall of the text blocks at the pixel level is around 98.74% in Saint Gall and 94.24% in the Parzival corpora.

First of all, a specialized classifier for the MBA is required since the RLSA cannot estimate accurately the MBA. The results draw a high Pixel Hit Rate in both corpora since the majority of pixels of the lines are well classified. However, on the Saint Gall dataset, the TLL-DR and MS-FM are a bit worse (the higher the better). We have noticed that in the Parzival dataset, the text is more comprised and it is easy to find the MBA, while in Saint Gall some regions are not fully covered by MBA classification. Nevertheless, the results on the historical datasets are very competitive, outperforming some of the state-of-the-art approaches, in particular on the Parzival dataset, as shown in Table 5.9.

5.10 Summary

TLE is one of this PhD Thesis main contributions. We have explored different bottom-up approaches, but always trying to extract useful text information by ML techniques. This information has been helpful for later segmentation/aggregation stages that not always relied on machine learning approaches, but that took advantage of the received information.

Examples of this, are, for instance, the joint work for the Combination and Optimization Problem with IP which proposed a flexible bottom-up text line extraction method for handwritten historical documents which was presented in [Pastor-Pellicer, Garz, et al. 2015]. Machine-learned interest points have been classified and grouped into text lines using combinatorial optimization; finally, a clustering step rejects invalid line candidates. Employing interest points and combinatorial optimization facilitates processing of text lines, blocks, and characters with arbitrary orientations and curvature. The advantage of learned IP over those extracted by geometrical heuristics, such as Difference-of-Gaussian, lies in the fact that knowledge about the nature of the interest points, for example, whether it is part of the text, noise, background or embellishment, is inherent. The IP have been classified, to determine their position on the segment (upper/lower contour or center of the text); thus, they define the limits of the segments, and provide information about where

to split touching components and touching lines. Approaching the problem of aggregating word segments to text lines using noise-robust combinatorial optimization, the method is capable of finding the globally optimal assignment of adjacent segments even for arbitrarily curved text lines. The key is the formulation of a suitable cost function; while favoring close-by neighbors, the cost function defined requires a text line to be smooth regarding orientation changes, i.e., sudden changes in direction are avoided and includes a penalty term for small segments, which are likely to be noise.

On the other hand, we proposed a MBA-based approach which avoids the extraction of IP since it directly classifies all the pixels of the image, published in [Pastor-Pellicer, Afzal, et al. 2016]. The pixel classification is done at layout level and then for the text line extraction. This method aims at historical documents as well since as we have seen in the previous model, the CNNs can deal with the typical historical artifacts which appear in those old documents and then, detect the line paths. The computation of the MBA makes the method robust against touching component Also, the classification is very robust due to it is performed at pixel level, and misclassification of one of the several pixels is smoothed by the rest of pixels, providing a high recall on the text line set. Mis-classifying one IP could have a bigger impact when determining the limits of the text segments. We have also empirically shown that the pre-processing Layout classification helps when the document presents decorations, annotations, and stains because they are ignored in this first stage.

Chapter 6

Text Line Normalization

Contents

Text Line Normalization (TLN) refers to the pre-process applied to each line before feeding them into the recognition engine. The purpose of TLN is twofold: to remove the variability associated with writers and writing devices (while preserving the relevant features for classification) and also to provide suitable inputs for the recognition engine. Most recognition engines transcribe the input lines once the normalized text line image has been transformed into a sequence of feature vectors. TLN aims to make the system invariant to the size of the characters, and to reduce the empty background areas caused by the ascenders and descenders of some letters. Figure 6.1

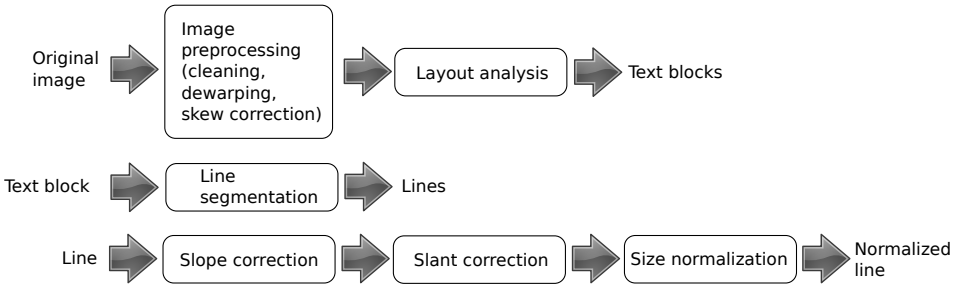


Figure 6.1: DIA pipeline. In this chapter we focus on the size normalization stage.

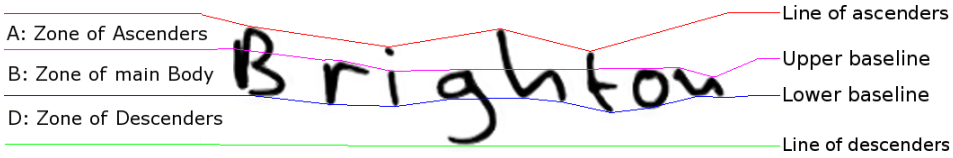


Figure 6.2: Example of text line image with the different zones (zone of ascenders, zone of descenders, and MBA) and the reference lines delimiting them (mean line and baseline, and the lines of ascenders and descenders) of the cursive script.

shows the general text line recognition pipeline. It shows the steps followed by our HWR engine, some of which could be skipped (depending on the corpora). TLN is on of the final pre-processing steps applied before the final recognition.

In previous works performed by the research group, a HWR system, based on HMM/ANN optical models was developed. Nevertheless, part of its success relied on a brilliant text line pre-processing. Indeed, the pre-processed lines have been recognized by other recognition engines, generally, improving their results [F. Zamora-Martínez, Frinken, et al. 2014]. The TLN methods followed by this system were based on detecting the text reference lines and then scale the parts of the text to a fixed size. A full explanation of the text reference lines is shown in the previous Chapter 5.2. These methods relied on the classification of LEPs into IPs as explained in Section 5.3. After classifying LEPs in one of the text reference lines class, it is straightforward to compute the final reference line by joining the points of the same class. An example is shown in Figure 6.2.

Though the normalization step by detection of reference lines by IP was successful, our intuition was that this stage could enhance the final transcription

outcome. The premise is that the classification of IP had some drawbacks that could make the normalization stage fail:

1. If an IP is misclassified, it could change the direction of the reference line for a large segment, and thus distort the content of the line.

2. If the line is short, or the goal is to normalize isolated short words. Then, there are not enough points of one class, and the text reference lines prediction become not possible.

We have to be careful with this issues the line can become unreadable. The primary goal of text reference lines is to reduce the variation of the text amongst lines, not to destroy them. Therefore, our goal in this chapter is to obtain a more robust method. Our way to overcome the mentioned issues consist on classifying all the pixels of the text line instead of specific IPs. The computation is now more exhaustive than previously, and therefore it requires efficient techniques.

6.1 Skew, slant and slope correction

In off-line HWR, the text line pre-processing, relies typically on slope and slant correction, and normalization of the size of the characters. Previous page level pre-processing steps should have been applied as well, like cleaning and skew detection (or TLE).

Skew Detection The skew detection and correction relies on the detection of the skewed angle [Hull 1998]. Usually, the skew is corrected at page level since all the page presents a similar skew, another approach is to apply a different skew correction to various text blocks.

Slope Correction With the slope correction, the handwritten word is rotated such that the lower baseline is aligned to the horizontal axis of the image. Note that if the line is skewed, we could use the slope correction to correct the skew as well.

Deslanting The slant is the clockwise angle between the vertical direction and the direction of the vertical text strokes. Slant correction transforms the word to an upright position. Ideally, the removal of slope and slant results in a word image independent on such factors. In this chapter, we rely on the fact that slope correction, slant removal, and text size normalization stages can be easily performed once the reference lines have been correctly tracked.

6.2 Text Line Height Normalization

Most of the HWR systems comprise the detection of the different zones of the cursive script: the Main Body Area (MBA) (between the mean line and baseline), the zone of the ascenders, and the area of the descenders. Besides, it allows to reduce the text line variance, and by detecting the baseline of the text, it is possible to correct the word slope. Not surprisingly, TLN approaches that are not limited to obtaining a rough estimate of the text baseline systematically improve recognition results. After delimiting the text zones, each of them is scaled to a fixed height by using an image interpolation algorithm. Each zone is assigned a portion of the total height. The MBA should have a larger extension than the other ones since it comprises most of the relevant word information. Regarding ascenders and descenders, one extreme approach is to code only with one pixel each of these zone, In this case, the ascenders and descenders properties of a column are binary features which indicate whether ascender/descender is present. Those features can discriminate e.g. between “b” and “a” or “o” and “q”. Nevertheless, it could have problems to distinguish between a “q” and “g”. Therefore, we will add a small amount of pixel for this zones. In the experiments carried out, ascender zone is reduced to 20% of the final image height and the descender region is reduced to 10%, producing a fixed height image suitable for the feature extraction step. The image height has been set to 42 (8-30-4) pixels as in [Espana-Boquera et al. 2011].

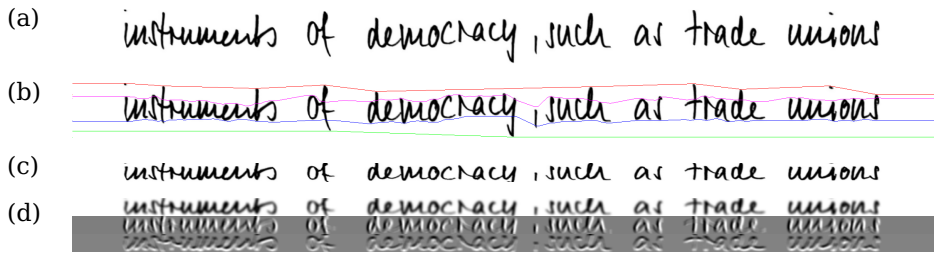


Figure 6.3: Example of a IAM text line image and height normalization. From top to bottom: (a) original image, (b) image with reference lines computed with the proposed technique, (c) image after text size normalization, and (d) visualization of the features extracted.

6.2.1 Column resizing model

There are several possibilities for the image resizing:

Independent Column model Since only height normalization is performed in our approaches, the re-sizing is applied treating each column independently. Each column is seen as 1D vector which is split into the 3 areas and is resized independently. This approach is the one applied in the methods presented in this Chapter. Note that in this case, the size normalization process does not preserve the aspect ratio of the images.

Fixed size In this case, the reference lines are scaled to a certain height for all the line image. These heights are computed as the average of each text reference line. This method performs well if the MBA is stable along the line, but it is not appropriated in the case of word size variations within the same line. As an advantage, it avoids significant change between columns.

Keeping Aspect Ratio Each column is computed independently from the rest, but the image as a whole is scaled preserving its aspect ratio. For each text column, we compute the aspect ratio as $\frac{|s_l - s_u|}{|d_l - d_u|}$, where s_l and s_u are the lower and upper reference lines from the original images (source), and d_u and d_l are the upper and lower reference line values for the normalized line (note that these values are constant throughout the line). The lines are then translated to the center of the line and scaled preserving the aspect ration. Note that some images may lose some parts of ascenders and descenders, although it is very unlikely to lose parts of the MBA.

A few remarks about the image scaling followed. As mentioned, the scaling is performed in one dimension (i.e. column by column), so we could use different interpolating methods. We used a simple bilinear interpolation. Though this may not be the best scaling method available, we obtained clear and smoothed normalized images prior the recognition.

6.3 HMM column model

Our first attempt to improve the IP-based normalization relies on the use of Hidden Markov Models hybridized with ANNs (HMM/ANN) which are applied column-wise in order to segment each column of the handwritten line into 3 zones. Our primary goal is to find a way to normalize handwritten text lines based on a supervised statistical model which takes into account all pixels instead of just local extrema. As we stated, the IP model degenerates as soon as points are misclassified. As the original IP-based TLN methods, this model also tracks the text reference lines, but these are obtained in a different way. Instead of joining IPs to generate the reference lines, the pixels are classified into the different zones. Thus, the reference lines are the frontiers between zones. In short, we calculate the text areas, and then we extract the reference lines as the boundaries between them.

6.3.1 Statistical framework

The zone detection problem can be formulated as a joint pixel classification problem into three classes $\{A, B, D\}$ for the zones of Ascenders, Main Body Area (MBA), and Descenders, respectively. This classification shows some restrictions: if we focus on a given column, the pixels of the same zone are contiguous, and the classes follow a vertical order (from top to bottom: A, B and D, as illustrated in Figure 4.5).

The zone estimation, posed in this way, can be readily formulated as a statistical pattern recognition problem. Especially, if this process is applied column-wise, the problem is a joint classification of sequences, which can be tackled using HMMs or CRF, since both provide the capability of combining syntactic restrictions with the estimation of likelihoods of each pixel given some features.

Formally, given an image of width w and height h, each one of the w image columns can be described as a sequence of pixels $X = (x_1 \dots x_h)$ and, under the statistical approach to pattern recognition [L. Rabiner et al. 1993], the

goal is to find the likeliest sequence of zones $Z^\star = (z_1 \dots z_h)$ that maximizes the a-posteriori probability:

$$Z^\star = \underset{Z \in \{A,B,D\}^h}{\arg\max} \; P(Z|X) \; . \tag{6.1}$$

The application of the Bayes rule leads to a decomposition of $P(Z|X)$ into the model $P(X|Z)$ and the statistical model that describes the apriori probability of zones $P(Z)$. The problem can then be reformulated as:

$$Z^\star = \underset{Z \in \{A,B,D\}^h}{\arg\max} \; P(X|Z)P(Z) \; . \tag{6.2}$$

A summarised explanation of the HMM/ANN can be found in Appendix B. The ANN estimates the posteriors $P(z|x)$, being x a set of features associated with the pixel to be classified. Therefore we need to convert to emission probabilities $P(x|z)$ by applying the Bayes rule:

$$P(x|z) = \frac{P(z|x)P(x)}{P(z)} \; . \tag{6.3}$$

The class priors $P(z)$ are estimated in the relative frequencies of each class from the training data. The HMM transition probabilities are estimated from the same data by Viterbi alignment. Note that, in this case, it is not necessary to perform an expectation maximization procedure: the re-segmentation step is not required since the artificial ground truth is labeled at the pixel level and we are dealing with a joint classification at this level. These scaled likelihoods can be used as emission probabilities ($P(x|z)$) since the scaling factor $P(x)$ is a constant for all classes.

6.3.2 Column HMM/ANN Modeling

A very simple left-to-right with loops HMM topology, as depicted in Figure 6.4, is sufficient to model the a priori probability of zones given by the constraints on the possible sequence of zone labels. Each one of the three emitting states corresponds to one of the three zones. The fact that some lines do not have ascenders or descenders is modeled by allowing their respective states to be skipped.

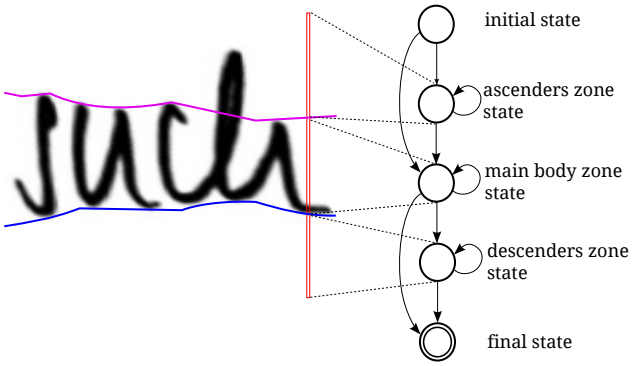


Figure 6.4: Scheme of the used HMM topology: one state for each zone $\{A, B, D\}$ and skips for ascenders and descenders.

The state emissions are estimated with an ANN that receives a centered window around the pixel to be classified in one of the 3 zones. To this end, the softmax activation function is applied to the MLP output layer. An alternative approach consists of determining whether the pixel belongs to the MBA or not, without discriminating between ascender and descender zones. In this case, the emissions of the states A and D are tied, and a single logistic output neuron is sufficient. For each column, we will obtain two points that delimit the transitions of the HMM (ascenders-MBA-descenders).

6.3.3 Adding the rest of Text Reference Lines

The HMM/ANN models that are described above are applied to each column of the image to segment it into 3 parts. White-space columns are skipped during training and pre-processing.

Note that the HMM/ANN only detects the MBA and therefore the upper and lower baselines, but not the ascenders and descenders limits. For this purpose, we use an approach similar to the IP-based method, but only LEPs that are outside of the MBA are taken. Then, the local maxima of the upper contour are used for ascenders and the local minima of the lower contour for descenders.

In addition some restrictions are applied:

- LEPs must be more than n pixels off the MBA. (n has empirically been set to 5 pixels)

- Only one LEP of each class can appear in the same column. In that case, the furthest point from the MBA is taken.

The final reference lines are determined by joining ascenders and descenders respectively using linear interpolation as can be seen in Figure 6.3.

Finally, once the reference lines have been obtained, normalization is performed for each column of the image by linearly scaling the 3 zones to a fixed height.

6.4 A combined Convolutional Neural Network and Dynamic Programming approach

In the results section (6.6) we will see that our first approach did not outperform the original normalization. Although it was able to solve some of the problems posed by the IP method, i.e. reducing the impact of misclassified points. A flaw was the assumption that the segmentation of pixel columns was independent, even though they share the same HMM/ANN model and part of the receptive field that is received by ANN. Smoothing was applied to the lines to avoid significant changes, but this was not sufficient to provide enough correlation between adjacent columns. In some cases, the reference points had significant changes between neighboring columns leading to undesired artifacts.

The following solution follows a similar procedure, we compute the reference line using DP on top of the ANN estimations, avoiding the use of HMMs. The DP algorithm used is based on Seam Carving [Avidan et al. 2007] and it seeks the reference line continuity constraints on adjacent columns in an explicit way. For this purpose, we applied a CNN that is trained to classify pixels as belonging to MBA. Meanline and baseline are obtained by searching paths of maximal energy applied over an energy map which is computed from a pixel-wise classification by the CNN. The received reference lines are used to normalize the text line images in the same way as the previous method and the original normalization. Figure 6.5 illustrates the workflow that is followed to normalize one line.

On top of the MBA estimation, we compute its vertical and horizontal gradients. Then DP pathfinding calculates the path with the highest score subject to some continuity constraints.

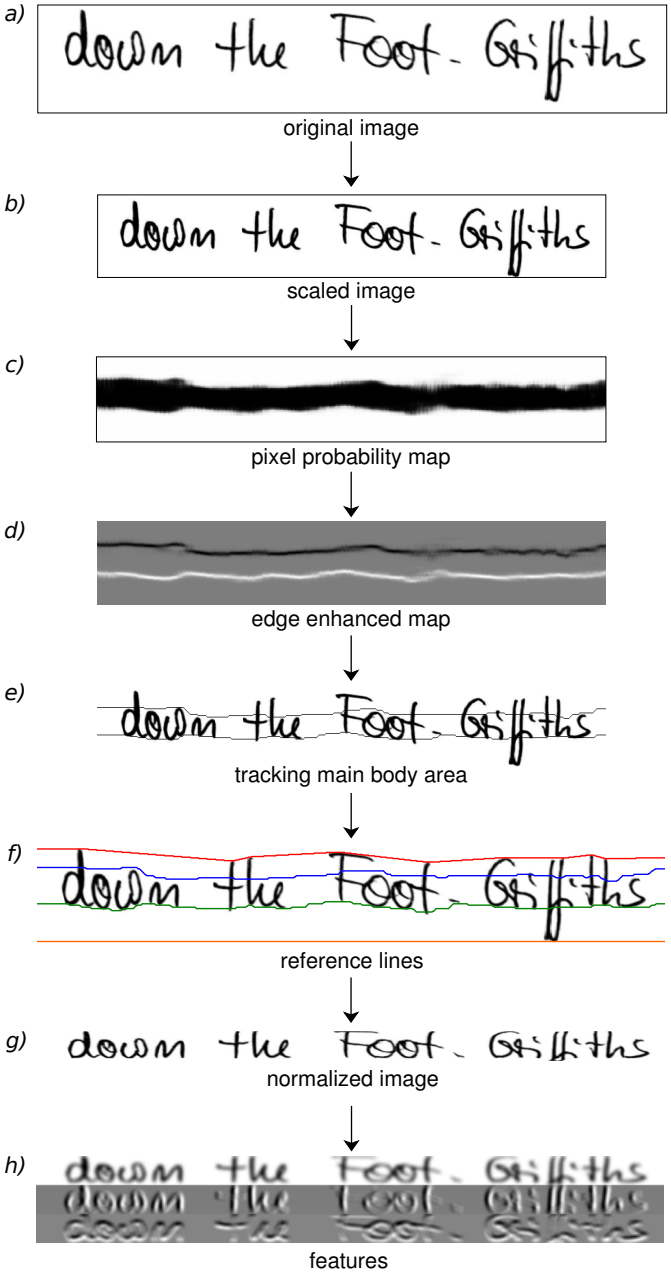


Figure 6.5: Text line normalization workflow by DP.

6.4.1 Pixel probability map

The purpose of this first step is to estimate, for each pixel of the text line image, the likelihood of belonging to the MBA (Figure 6.5-c). Here, we apply the one-to-one pixel labeling explained in Chapter 3.7. Indeed, the classification task is the same as the one presented in chapter 5.6 but at line level instead of on the whole page. CNNs can compute features from a window by applying convolutional kernels and sub-sampling layers. Different configurations have been tested considering the possibility of labeling either one pixel or one column at a time. The best figures of merit correspond to the second approach which is illustrated in Figure 6.6.

In that case, the input of the net is a full column and its surrounding context. This column-wise labeling requires text line images to be scaled to a fixed height (Figure 6.5-b). This may seem paradoxical since one of the purposes of height text normalization is to provide a fixed height for all the images. In this case, we need a fixed height to detect the MBA to normalize the image to a fixed height, although in the first resizing the images are scaled keeping their aspect ratio.

For this first scaling[1], the images are scaled to a height that is big enough to keep the properties of the image. This height has been obtained by analyzing the size of each area on the training set, the averages for ascenders, MBA, and descenders are, respectively: 48, 30 and 44. Hence the image are scaled to 122 pixels, Then we reduced the whole images by a factor of 2 before applying the column CNN. Therefore, the associated sizes of these areas are 24, 15 and 22. To this end, a rough estimate of the reference lines is obtained using the horizontal projection histogram. The generated text reference lines correspond to the first line exceeding 50% of the maximum value from the bottom and top, respectively. The images, then, are scaled vertically, translated and finally cropped. Some scaled images may loose some parts of ascenders and descenders, although it is very unlikely to loose parts of the MBA.

[1]Note that this is the fixed size that we use to apply the column-wise MBA estimation model. Then, after using the proposed normalization approach, the line will be reduced to 42 pixels. The first scaling keeps the aspect ratio of the whole image, while in the second stage each column has a different aspect ratio for each of the areas.

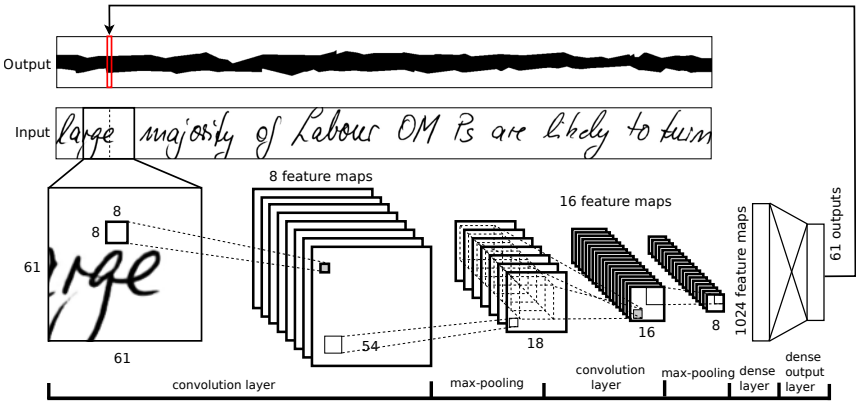


Figure 6.6: Process to obtain the pixel probability maps from the scaled images by using a CNN. The input is a window centered in one column, while the output is the probability of all the pixels of this column to belong to the MBA.

6.4.2 MBA edge detection

After estimating the probability map by means of CNNs, the next step for computing the reference lines is to highlight the edges enclosing the MBA. This leads us to an energy map as illustrated in Figure 6.5-d. A discrete differentiation operator is used to this end, by using the following filters which estimate the vertical gradients for the upper and lower contours, respectively:

$$\text{upper} = \begin{bmatrix} 2 & 2 & 2 \\ -1 & -1 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \text{ lower} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & -1 & -1 \\ 2 & 2 & 2 \end{bmatrix} \tag{6.4}$$

Note that this filters are similar to Sobel but they have been modified to detect upper and lower edges separately.

6.4.3 Reference lines extraction by Dynamic Programming

We convolve the previous filters separately, generating 2 "energy maps" regarding to the text meanline and baseline. Both should be continuous left-to-right paths. The path of maximum energy subject to this constraint can be computed using DP as in Seam Carving algorithms. In these algorithms, the continuity constraint can be modeled by limiting the search to paths with just one row per column and restricting the distance between consecutive rows.

In this work, only adjacent rows are taken into account to compute the score f for each pixel position (x, y) from the corresponding energy map e. The recursive equation for the general case is formulated as follows:

$$f(x,y) = e(x,y) + \min \begin{cases} f(x-1, y-1) \\ f(x-1, y) \\ f(x-1, y+1) \end{cases} \tag{6.5}$$

The remaining text reference lines (ascenders and descenders) are added following the procedure described in Section 6.2.1.

6.5 Experimental Setup

Both of classifiers presented here (MLP in the HMM and CNN in the DP algorithm) estimate the MBA of the text line. Therefore, the same (soft) ground truth is used for training the networks of both approaches (Appendix E.2), which are obtained from the LEP classification technique used in [Espana-Boquera et al. 2011]. Hence, it is expected some mistakes that are inherited from it. Nevertheless, we have observed that the proposed MBA estimation can recover some errors generated by the LEP classification.

6.5.1 HMM/ANN column model setup

For training the MLP we sampled the several hyper-parameters by using the Random Search Hyper-parameter Optimization technique discussed in Chapter 3.6. The explored parameters include:

- the learning rate and the momentum term are taken from a log-uniform distribution between 0.001 and 0.5,
- the loss function can be either chosen from cross-entropy and MSE,
- the weight decay is chosen from $\{10^{-5}, 10^{-6}, 10^{-7}\}$,
- the size of the first hidden layer is chosen from $\{64, 126, 256, 512\}$ and the second hidden layer from $\{16, 32, 64, 128\}$, with the restriction that the first must be greater than the second,
- the minibatch is sampled from $\{16, 32, 64, 128\}$,
- the window size to model the input pixel window: the width and the heights are chosen independently to be $2 \times n + 1$ pixels, for $n \in \{10, 20, 25, 30, 35, 40\}$,

Layer	Type	Kernel Applied	Output
1	input	$1(61\times61)$	
2	convolution	$8(8\times8)$ kernels	$8(54\times54)$
3	max pooling	3×3	$8(18\times18)$
4	convolution	$16(3\times3)$ kernels	$16(8\times8)$
5	flatten		1024 neurons
6	fully connected (relu)		256 neurons
7	Output (logistic)		61 neurons

Table 6.1: Topology of the CNN for MBA estimation.

The best configuration for this experimental setting was:

- the input layer receives a window of pixels of width $2 \times 35 + 1$ and height $2 \times 35 + 1$ centered at the pixel to be classified as well as the vertical distance to the upper and lower the vertical contour of columns of a window of the same width,
- two hidden layers of sizes 256 and 64, respectively,
- learning rate 0.15, momentum term 0.2 and weight decay 10^{-6}

Note that HMM/ANN is applied in this step to downsized images to reduce the input parameters of the net and to speed up the process. The images were downsized to 50% of their original size. After computing the mainline and baseline, these are scaled back to the original size. We have also observed that applying the technique every other column produces nearly the same results.

6.5.2 Combined CNN and Dynamic Programming

For this approach the classification performed by the CNN is very similar to the neural models presented in Chapter 5.6, but instead of classifying one pixel at a time we compute the total pixels of one column. The output of the network have now 61 logistic units; each one represents the probability of the pixel x_i to belong to the MBA.

As depicted in Figure 6.6, the input of the net is a 61×61 squared window. Despite the image downsampling, this window is big enough to label the MBA of the central column reliably. After some scanning of topologies, the chosen CNN is described in Table 6.1.

System	$\|\Omega\|$	WER (%)	CER (%)
MLP Normalization (HMM/ANN)	103K	21.1	8.6
HMM Zone estimation (HMM/ANN)	103K	24.4	10.6
CNN+DP Normalization (HMM/ANN)	103K	19.0	7.5

Table 6.2: WER and CER for the test set of IAM database.

In this case, the CNN has been trained with BP and adadelta ([Zeiler 2012]) as optimizer, in combination with regularization methods such as weight decay penalty, and Gaussian noise on the CNN input.

6.6 Results

To assess the performance of the methodologies presented in this chapter we have used the IAM Offline Database for our experiments (Appendix C.3. We applied a recognition engine based on Hidden Markov Models hybridized with ANNs (HMM/ANN) which is detailed in Appendix B. Pre-processing steps (all but height normalization) are fully explained in [Espana-Boquera et al. 2011]. For the LM we used a vocabulary of $103k$ words and 4-grams as described in Appendix B.3.

Table 6.2 shows the text line recognition results that were obtained for the test set together with our baseline (MLP Normalization). A complete table has been included in next chapter (Table 7.4, including the following works using HMMs and CNNs for features extraction, as well as, other relevant works in the evaluated data. Careful attention must be paid when comparing with other works even if they use the IAM database and the same training and test partitions, since the lexicon, the language models, and other parameters may vary. The metrics reported as usual in handwriting recognition tasks are Word Error Rate (WER) and Character Error Rate (CER).

Table 6.2 show the comparison of different line normalization methods in this context of the same recognition engine, only by using different line normalization techniques while fixing the other HWR components. Table 6.3 show more detailed comparison of the two methods developed in this section.

The HMM Zone estimation model showed worse results than the original normalization . Nevertheless, we have observed that the proposed area estimation method can recover from some errors generated by the local extrema classification technique. For example, as illustrated, in Figure 6.7 the capital

	Validation Set		Test Set	
System	WER (%)	CER (%)	WER (%)	CER (%)
HMM Zone estimation	18.6	6.9	24.4	10.6
CNN+DP Normalization	13.9	4.5	19.0	7.5

Table 6.3: Word Error Rate (WER) and Character Error Rate (CER) for the test set of IAM database.

(a) (b)

Figure 6.7: Example of IAM image normalization: (a) zone detection by applying HMM/ANN, (b) detection of the reference lines LEPs.

"D" got a weird shape becuase of missclassified IP (left figure). By counterpart, the CNN+DP model outperforms the other two models improving the error in almost 2 absolute points in CER and the test set (5% relative improvement).

6.7 Discussion

Two different methods for text line size normalization have been presented, discussed, and evaluated. Even though the first approach (HMM Zone Estimation) does not outperform our initial model, its underlying ideas there have been extended for the CNN+DP model, which actually improved the previous and the original models significantly.

The estimation of the different zones are computed pixel-wise by applying an MLP and CNN (respectively) to each column of the text line image. Then this information is combined with a HMM in the first case and DP for the second approach. Nevertheless, the application of the proposed methods in this work for slope normalization is straightforward, we just need to translate each column to the same height on the baseline.

It is worth it to remark, to the best of our knowledge, this is the first time that CNNs are used to track the reference lines of handwritten text line images.

6.8 Summary

After analyzing the errors performed by our recognizer, we noticed that some errors were due the text line normalization stage. Specifically, due to the misclassification of IP while extracting the text reference lines. As expected, by improving this stage it was possible to improve the overall recognition. We have presented two approaches that are based on classifying pixels of the image instead of LEPs/IPs. The first approach did not show an improvement in performance. However, the second approach did finally improved the recognition results. Both approaches are published in [Pastor-Pellicer, España-Boquera, Castro-Bleda, et al. 2015; Pastor-Pellicer, España-Boquera, P. Zamora-Martínez, et al. 2014].

Chapter 7

Decoding from Scratch

Contents

In the previous chapter, we made use of our Handwriting Text Recognition recognizer based on Hidden Markov Models hybridized with ANNs to evaluate the impact of our Text Line Normalization methods. Normalized images are followed by a feature extraction stage before applying the HMM/ANN decoder. However, connectionist methods and especially deep neural networks are able to extract meaningful features from raw values (in off-line HWR, text images). More specifically, this chapter proposes the use of deep ANNs (MLPs and CNNs) for this purpose. CNNs have already been used in this PhD Thesis in several DIA stages: Document Image Binarization, Text Line Extraction, layout and Text Line Normalization. The performance of the HWR engine using deep learning and CNNs to extract meaningful features has been impressive, as the experiments will show.

In our baseline HWR system [Espana-Boquera et al. 2011], the emission probabilities are estimated by an MLP, whose input is a sequence of feature vectors following [Toselli, Juan, et al. 2004]. An illustration of the baseline system is depicted in Figure 7.1. In the presented approaches, instead of relying on a previous feature extraction process, we rely on the raw image input, by using a deep neural network to directly extract meaningful features (see Figure 7.2) or by using CNNs (see Figure 7.3).

In section 7.2 we introduce the models based on deep MLP for feature extraction and in Section 7.3 the convolutional topologies used for this purpose. The experimental setup and results are described and analyzed in Sections 7.4 and 7.5.

7.1 Previous feature extraction and receptive field

In the feature extraction procedure proposed by [Toselli, Juan, et al. 2004], 60 features per frame were extracted (3 features from 20 cells with 2.1 pixels step size). In the models trained in the previous chapter, the ANN received a context of 11 frames which makes a total of 660 input values. The feature extraction for each HMM step cover a 27×27 pixel window as depicted in Figure 7.4:

1. We start with an MLP that receives 660 features as input.
2. These features are taking from M frames. Each column has 60 (3×20) features. The number of frames used is $M = 660/(3 \times 20) = 11$.
3. Each of these cell values is taken with an advance step of 2.1 columns on the original image. Hence these frames cover $2.1 \times M \simeq 23.1$ columns on the original image.
4. Then we have to take into account that the values of each column have been taking by a 5×5 feature extractor cell. It involves the use of 2 extra padded columns on each side of the cell. After adding these padded values, the final effective input for the MLP is ~ 27.1 wide pixels.

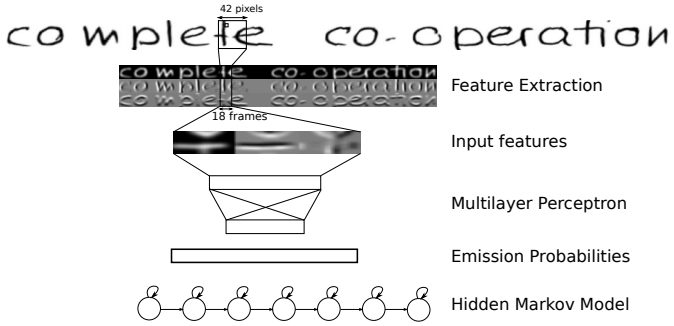


Figure 7.1: Baseline HMM/ANN recognition system using a sequence of feature vectors as input.

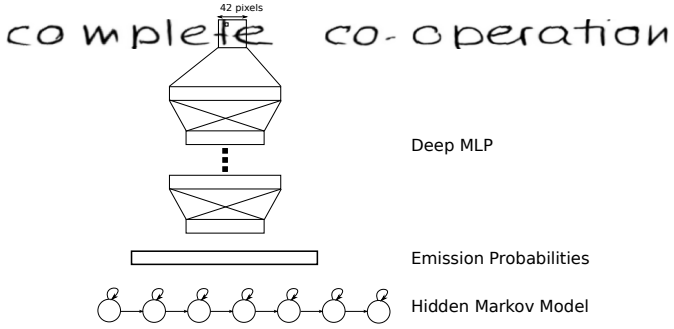


Figure 7.2: HMM/ANN recognition system using the raw image as input.

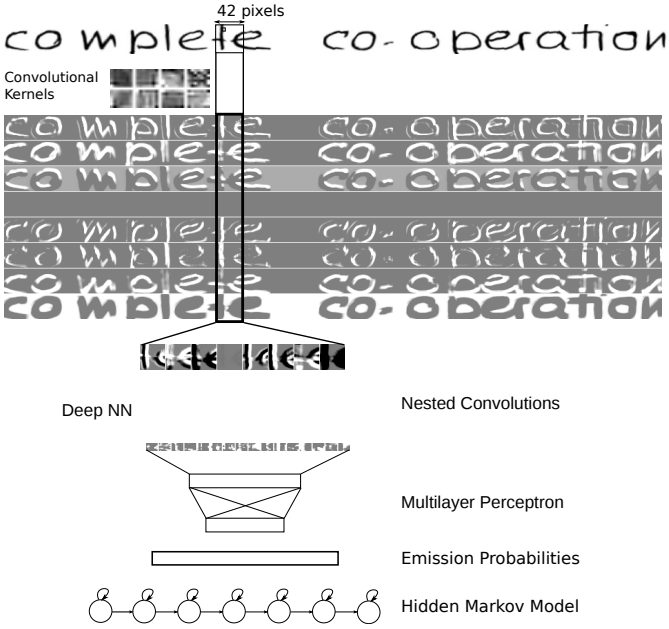


Figure 7.3: HMM/CNN recognition system.

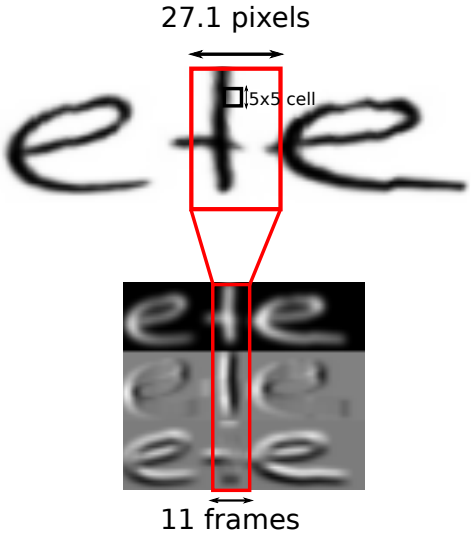


Figure 7.4: Effective image window on the MLP window.

7.2 Deep MLPs for extracting features

We aim at extracting meaningful features for HWR using deep learning techniques. When using the baseline system, the input of the MLP is a set of feature frames that are centered at the current frame. In the new architecture, the sliding window is a patch of raw pixels that are fed directly to the ANN. The choice of a squared window has given good results in preliminary experiments, leading to a window of 42×42 pixels. The window advances two pixels at a time. The overall architecture is similar to the one presented in [Pastor-Pellicer, España-Boquera, Castro-Bleda, et al. 2015], but, in this case, the raw image is used as input instead of the computed features.

When dealing with raw images, there are several issues to keep in mind to improve the performance and generalization. Several standard regularization methods such as weight decay or max weight penalty have been employed. The input of these deep models is a 1D vector; therefore, adjacent windows have different representations due to translation. Regularization techniques such as dropout have helped to improve results. We have also tried a layer-wise pretraining with SDAE [P. Vincent et al. 2010] in order to train deeper nets.

7.3 CNNs for preprocessing

In the classical HMM/ANN architecture, the use of a sliding window (where the same ANN is applied to classify each frame) can be seen as a 1D convolution on the X axis. In addition, we would like to explore the use of 2D convolutions combined with pooling layers and higher level convolutions that will hopefully be able to extract more useful features.

In our setup, unlike fully convolutional networks [Long et al. 2014], each window is treated as an independent classification problem, in spite of the fact that the CNN is applied to the whole text line. It is an imperative and challenging task to obtain an architecture with a good cost-efficiency trade-off.

Figure 7.3 illustrates the CNN for feature extraction and conditional probability computation. In the proposed settings, several parameters must be chosen for the CNN, such as the number of convolutions, pooling layers, activation functions, number, and size of the convolutional kernels as well as the classifier, which usually is an MLP. For this choice, the computational restrictions

that are essential for finding an appropriate but efficient architecture must not be forgotten.

We have explored three alternatives with all of those limitations in mind, namely: using known CNN architectures, using a specific network for the mentioned task, and using a model inspired by a well established feature extraction technique.

7.3.1 Using known architectures

Our first attempt using CNNs for feature extraction imitates some of the previous architectures that have achieved good results in similar tasks. This is the case of the convolutional net LeNet CNN Lecun et al. 1998, which obtained good results on the MNIST database. In addition, the increase in computational resources (especially advances in GPU computing and distributed systems) has allowed the use of deeper and more complex models. In recent years, these issues, combined with an appropriate parameter tuning, have led to remarkable improvements in performance, especially in image vision tasks. This is the case of nets like AlexNet [Krizhevsky et al. 2012], GoogleNet Szegedy et al. 2014, and Very Deep Convolutional Networks [Simonyan et al. 2014], which have reported excellent results in other tasks such as the ImageNet Large Scale Visual Recognition Challenge contest [Russakovsky et al. 2015]. Nevertheless, our task still requires more resources than traditional image object classification since each convolutional forward is applied for each position of the sliding window, even though we have to deal with somewhat smaller inputs. In addition, it is hard to apply models with many subsampling or pooling steps since the input is not big enough.

7.3.2 Adhoc networks dealing with the singularity of HWR

Most of the architectures found in the bibliograpy are designed for tasks like MNIST, which consists of 28×28 pixel images corresponding to the 10 digits, or, for instance, the ImageNet database, which has larger inputs and more than a thousand classes. However, in our case study, the net input consists of 42×42 pixels, and there is an output for each different HMM state, which corresponds to 553 neurons (7 states $\times$ 79 graphemes).

When tuning a ANN model, in an ideal case we have to explore, every possible parameter and hyper-parameter in order to obtain the most successful configuration. The use of CNNs and deeper nets makes this tuning process worse since more parameters are added, most of which are related to the new

topology and layer configurations. Thus, in order to guide our exploration, we should concern about the kernel sizes to extract useful features, the number of kernels to cover the variability of the text, and deeper layers of the model to properly represent the characteristics of the problem.

Therefore, in order to guide our exploration, we should ask ourselves some questions about the net configuration such as the following ones:

- **Kernel sizes**: *What size of kernel can extract useful features from handwritten text (borders, shapes,...)?*
- **Number of kernels**: *How many characteristics must be extracted from a window to cover most of the handwritten styles and variations?*
- **Deeper layers of the model**: *How could these features be properly combined to represent handwritten characters?*

In the feature extraction process proposed in [Toselli, Juan, et al. 2004], the frames are computed using 5×5 cells. Coincidentally, *LeNet-5* uses 5×5 convolutional kernels in both convolution layers. We will, therefore, explore kernel sizes between 5 and 8 pixels per side allowing the model to consider slightly bigger window sizes.

When analyzing the kernels trained in some preliminary experiments, we could conclude that there is a tendency to extract redundant information from 16 kernels in the first convolution. Some of the learned kernels detect edges in several orientations, others estimate the ink text zones. It turns out that all of these features can be extracted with no more than 5 to 10 kernels. Due to the above-mentioned computational constraints, we will avoid the use a large number of kernels, at least, in the first convolutions.

7.3.3 Cell feature extraction by convolutions

Our baseline HWR system used the parametrization described in [Toselli, Juan, et al. 2004]. In this work, we will design CNNs that are powerful enough to mimic this feature extraction process. However, it is important to note that the convolution kernels are not limited to extracting these features since they will learn their own discriminative features.

The original feature extraction divides the input into cell regions. For each region, three values are extracted: one value with the proportion of gray level

Figure 7.5: How the 3 features extracted per cell will look if they were approximate by a convolution kernel.

in the cell and two values for the vertical and horizontal derivatives. A linear regression model is performed to find the optimal derivative directions.

This operations could be coded as convolutions (without hardcoded weights), for example for a cell size of 5×5, they will look as:

$$\text{normalized gray levels} = \begin{bmatrix} \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \\ \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} & \frac{1}{25} \end{bmatrix}$$

$$\text{vertical} = \begin{bmatrix} \frac{1}{20} & \frac{1}{20} & \frac{1}{20} & \frac{1}{20} & \frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & \frac{1}{20} & \frac{1}{20} & \frac{1}{20} \\ 0 & 0 & 0 & 0 & 0 \\ -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} \\ -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} & -\frac{1}{20} \end{bmatrix}$$

$$\text{horizontal} = \begin{bmatrix} \frac{1}{20} & \frac{1}{20} & 0 & -\frac{1}{20} & -\frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & 0 & -\frac{1}{20} & -\frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & 0 & -\frac{1}{20} & -\frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & 0 & -\frac{1}{20} & -\frac{1}{20} \\ \frac{1}{20} & \frac{1}{20} & 0 & -\frac{1}{20} & -\frac{1}{20} \end{bmatrix}$$

One convolution computes the vertical derivatives from the differences between the upper and lower cell values, similarly, another convolution can compute the horizontal derivatives, whereas a third estimates the smoothed gray level of the cell (Figure 7.5).

With this ideas in mind, we can use totally learned kernels (convolutions) that can perform a feature extraction in a similar way.

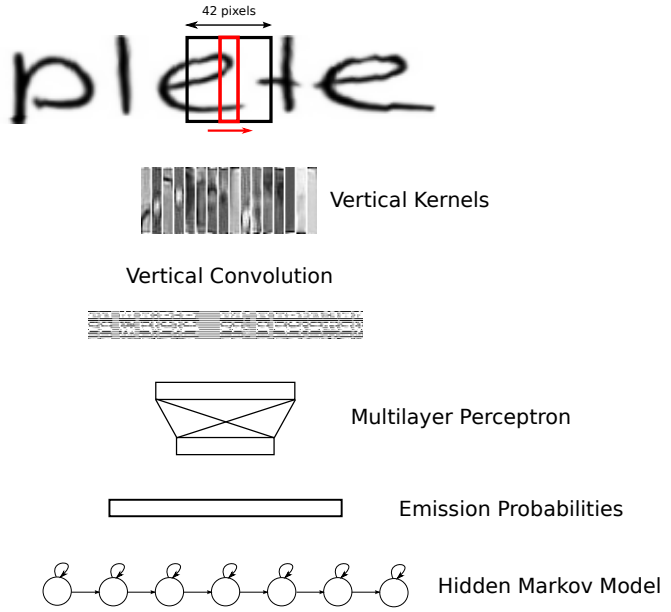


Figure 7.6: Vertical model (I). The kernels run over the input window and only in the horizontal direction.

Hidden Layers	Dropout (droprate)
2 hidden layers (2048, 512)	0, 0.2, 0.5
3×512 + SDAE	0, 0.2
5×512 + SDAE	0, 0.2
7×512 + SDAE	0, 0.2

Table 7.1: Deep MLPs fed directly with a raw image input from a window size of 42×42 (1764 pixels).

7.3.4 1D convolutions

We have also explored a CNN that convolves the text line images in only one direction. The convolutional kernels would have the height of the image, and they would advance from left to right. Therefore, each kernel extracts only one feature for each column. We explored two different approaches:

- Applying the vertical kernels directly into the raw image (Figure 7.6).
- Applying the vertical kernels after a 2D convolved map (Figure 7.7). In this case, the first set of 2D convolutions is obtained followed by the application of 1D kernels to these previous maps.

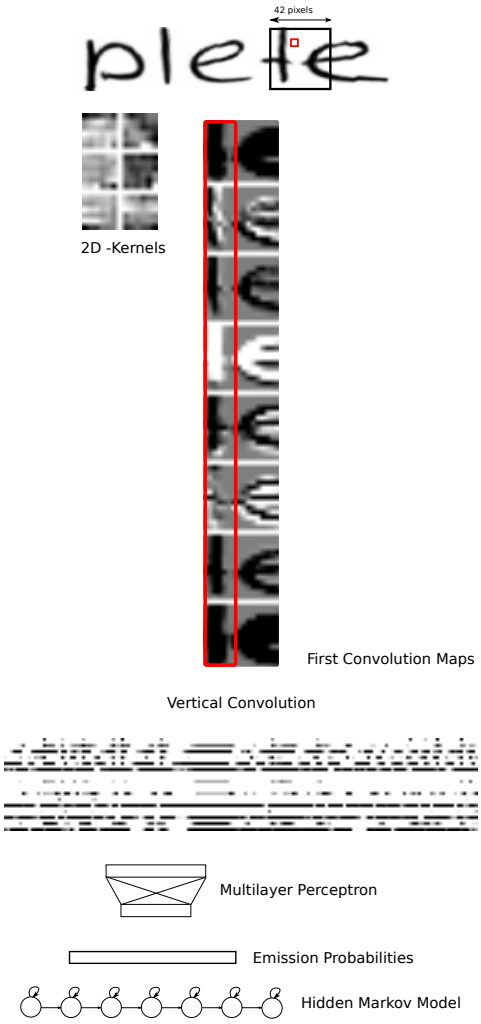


Figure 7.7: Vertical models (II). Vertical kernels run over the maps generated by the first 2D convolutions.

In the first case, the vertical kernels are applied to the input window, and we have a set of $K \times F$ features (with K being the number of kernels, and $F = C - w + 1$ where C is the number of columns of the sliding window and w is the width of the kernel since padding is not applied). In the second example, a 2D convolution is performed using the same parameters as in previous models. The vertical kernels are applied to the extracted maps afterward.

Although these 1D kernels are bigger than in the previous case, the model is still quite efficient since the convolution is applied only in one direction of the image. A larger number of kernels is used to overcome the restriction of having one feature per kernel and column. A drawback of this technique is that the system is less robust to distortions on the vertical axis such as vertical image translations. This issue is alleviated by the use of text line normalization approaches that reduce these vertical variations.

7.4 Experimental setup

In this section, we evaluate the performance of the proposed models. We show the configuration finally used and the result obtained for each one. Finally, we will visualize some of the kernels and the resultant maps of the proposed net architectures. The experimental framework and evaluation are the same than used in the past HWR and decoding experiments. The decoder applied, and setup can be seen in Appendix B but, as explained in this chapter, we removed the handcrafted parametrization step by using deep MLPs and CNNs. The data used for the recognition was preprocessed following the method explained in Section 6.4.3.

7.4.1 Using raw input and deep MLPs

Our first goal is to compare the baseline system with a new one, avoiding an explicit handcrafted feature extraction. Table 7.1 shows the configuration used for the deep MLP-based systems with a receptive field (42×42 pixels). We also trained deeper MLPs up to 7 hidden layers of 512 ReLU neurons that were pre-trained with SDAE in order to obtain faster convergence. The use of dropout helped significantly in these configurations.

7.4.2 Using CNNs for preprocessing

Table 7.2 summarizes the CNN topologies explored. First, a topology based on *LeNet* (*LeNet-5*) was tested. For the second alternative, after several trials, we could highlight one special configuration, called *Adhoc CNN*, which led us to the best results. We also decided to apply *max pooling* layers to not only speed up the computations but also to make our model more robust to translations. We tried increasing max-pooling layers of 3×3 and 4×4. Since the suitability of the max-pooling is very task dependent, we also performed experiments without them.

Finally, the models with the minimal configuration able to imitate the cell feature extraction were tagged as *Cell/Kernel*. As can be observed, the size of the kernels increased up to 6×6 and a stride of 3 was applied in each direction. Two different topologies with one and two convolution-activation-pooling layers were tried.

7.5 Results

7.5.1 Recognition results

The best results of the baseline system were presented in Pastor-Pellicer, España-Boquera, Castro-Bleda, et al. 2015, obtaining a 15.6% and 19.0% WER for validation and test sets, respectively. Table 7.3 shows the overall performance of the proposed systems, with a confidence interval of 95% [Vilar 2008]. First, it can be observed that all the deep models with more than two layers using raw inputs improved the baseline version. Indeed, when using two hidden layers in the raw setup, the results were worse than the baseline, unless dropout was added, where the results were similar (Figure 7.8). Dropout significantly helped in the deep model modality, reaching the best performance with three hidden layers and a drop rate equal to 0.2, obtaining a WER of 13.7 for the development set. We tried drop rates that were larger than 0.2 but the performance did not improve. As a matter of fact, although some results with deep models were better than others, there was no statistically significant difference among them. For CER, deep models statistically improve the baseline system.

The HMM/CNN also showed better performances with respect to the baseline system. When compared with the deep MLPs using raw inputs, the results were similar when dropout was used. When exploring the different nets, good

	Type	Kernel	Output
LeNet-5	input		1×42×42
	convolution	16 kernels 5×5	16×38×38
	Max-pool (ReLU)	2 ×2	16×19×19
	convolution	32 kernels 5×5	32×15×15
	Max-pool (ReLU)	2×2	32×8×8
	flatten		2048 neurons
	fully-conn. (ReLU)		500 neurons
	output (softmax)		553 neurons
Adhoc CNN	input		1×42×42
	convolution	8 kernels 7×7	8×36×36
	max-pool (ReLU)	3×3	8×12×12
	convolution	16 kernels 3×3	16×10×10
	max-pool (ReLU)	2×2	16×5×5
	flatten		400
	fully-conn. (Relu)		128
	output (softmax)		553
Cell/Kernel 1 Conv.	input		1 × 42 × 42
	convolution	8 kernels 6×6+3+3	6×13×13
	flatten		1014
	fully-conn.		512
	fully-conn.		128
	output (softmax)		553 neurons
Cell/Kernel 2 Conv.	input		1 × 42 × 42
	convolution	8 kernels 6×6+3+3	6×13×13
	convolution	16 kernels 4×4+2+2	16× 6×6
	flatten		576
	fully-conn. (ReLU)		256
	fully-conn. (ReLU)		64
	output (softmax)		553
Vertical 1	input		1×42×42
	conv. (vertical)	16 kernels 42×6+1+1+3	16×1×13
	flatten		208 neurons
	fully-conn. (ReLU)		256 neurons
	fully-conn. (ReLU)		64 neurons
	output (softmax)		553 neurons
Vertical 2	input		1×42×42
	convolution	8 kernels 6×6+3+3	6×13×13
	conv. (vertical)	16 kernels 13×4+1+2	16×1×5
	flatten		80 neurons
	fully-conn.		256 neurons
	fully-conn.		64 neurons
	output (softmax)		553 neurons

Table 7.2: CNN topologies for the recognition system.

			Dropout	Dev. WER	Dev. CER
HMM/ANN	Baseline			15.6 ± 1.1	5.6± 0.5
	Raw input	2048−512	0	16.1 ± 1.1	5.8 ± 0.5
	+ Deep MLPs		0.2	14.6 ± 1.1	5.1 ± 0.5
		3×512†	0	15.4 ± 1.0	4.9 ± 0.4
			0.2	**13.7 ± 1**	**4.7 ± 0.4**
		5×512†	0	14.1 ± 1.1	4.6 ± 0.4
			0.2	14.2 ± 1.0	4.9 ± 0.5
		7×512†	0	15.2 ± 1.1	4.9 ± 0.5
			0.2	14.5 ± 1 .0	5.1 ± 0.4
HMM/CNN	LeNet-5			14.6 ± 1.1	4.6 ± 0.4
	Adhoc			14.4 ± 1.1	4.8 ± 0.4
	Cell-kernel 1			**13.9 ± 1.1**	**4.4 ± 0.4**
	Cell-kernel 2			14.3 ± 1.1	4.9 ± 0.4
	Vertical 1			15.5 ± 1.1	5.2 ± 0.4
	Vertical 2			15.3 ± 1.1	5.4 ± 0.5

Table 7.3: Overall performance of the proposed systems on the Development set (configurations with the † mark make use of SDAE).

performances in the *Adhoc CNN* net or even *LeNet-5* could be expected. Even though the performance in these cases is quite good, the best result achieved so far has been with a simple net, using one convolution with a stride of three in each direction and only six kernels. We presume that the simplicity of the model eased the training, and with six kernels the model covers most of the variability of the handwritten text (as illustrated in Figure 7.12). In this particular case, the net extracts 1014 features from the convolution process, which are conveniently combined with two fully connected layers of 512×512, respectively. The *Adhoc CNN* model reduces the feature space to 400 after convolutions, compared with 2048 in *LeNet-5*. The model identified as *Cell-kernel 2*, which uses a second convolution level, had an fine performance that was not far from the best models.

Finally, we observed that the vertical models had a more modest performance, which did not improve the traditional 2D convolution models, but they were still better than the baseline. As before, CER was significately better with the HMM/CNN than with the baseline system.

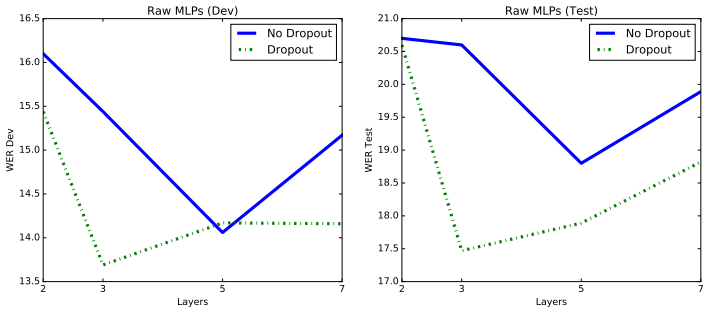


Figure 7.8: Evolution of the error by the number of hidden layers.

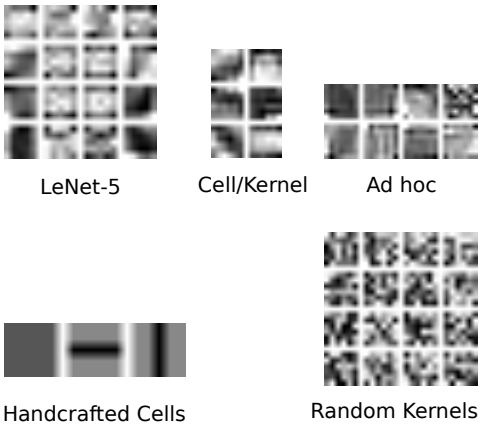


Figure 7.9: Weights of the different kernels in the first convolution for some configurations.

7.5.2 Kernel and map visualization

Let us analyze the shape of the learned kernels and maps to gain an insight about the features learned by CNNs and deep MLPs. Figure 7.9 shows the kernels learned by the convolutional models presented in this work. In addition, we plotted the weights of the first layer of one of the evaluated deep MLPs in Figure 7.10 (brighter pixels indicate higher activation values).

The learned kernels do not seem to be extracting vertical and horizontal derivatives, but rather locating edges and shapes. In fact, the kernel visualizations are not so illustrative if they are not accompanied with the maps that are generated by each filter, as shown in Figure 7.11.

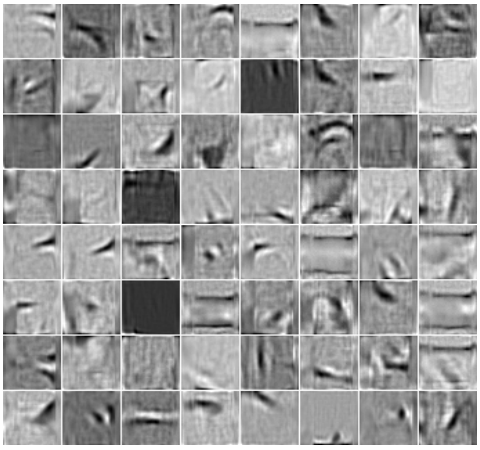


Figure 7.10: Weights of the first hidden layer of the deep MLP. Only a portion of the 2048 neuron weights is shown.

The most illustrative sample comes from the net identified as *Cell/Kernel* (Figure 7.12), which is the configuration that gave us the best performance. The figure shows the filtered maps by the six kernels of the first convolution. It is worth mentioning that redundant and irrelevant kernels could also be learned, as we observed when analyzing some of the maps generated by the different nets.

On the other hand, if we analyze some of the features extracted by the fully connected neurons in the deep MLP with raw inputs (Figure 7.13), each neuron detects a characteristic in a specific location of the input window. Thus, it requires several neurons to extract the same feature in different locations. This translation invariance is solved in the convolutional model, where kernels convolve the image by extracting one type of feature by the kernel in various positions. Thus, between $128 \sim 2048$ neurons were required to extract useful features, while similar or even better results were obtained with a few convolutional kernels ($6 \sim 16$).

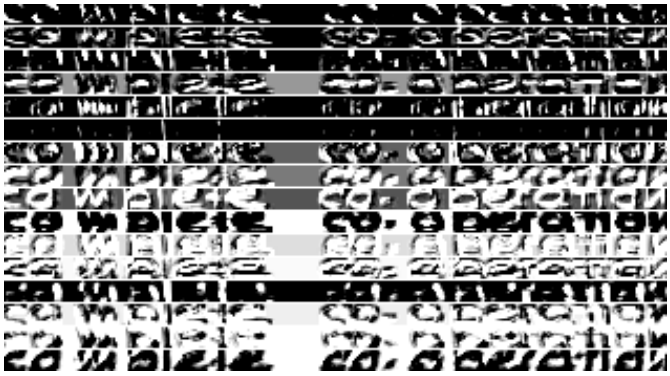


Figure 7.11: Maps extracted by the Adhoc CNN. For instance, one kernel generates lower/right contours (fifth kernel), and another learns the upper/left edges (fifth from the tail).

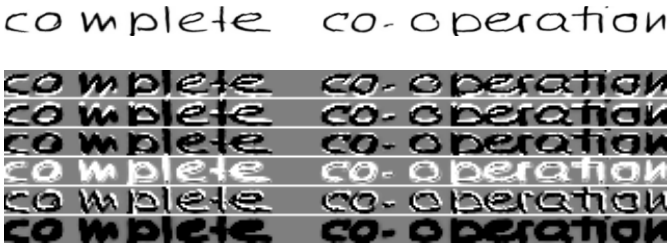


Figure 7.12: Six maps extracted by the first convolution of the *Cell/Kernel*. A free interpretation of the features learned is: 1) lower text contours, 2) upper contours, 3) borders, 4) strokes 5) right contours, 6) background model (background pixels got higher activation than text).

Figure 7.13: Generated maps by LeNet-5.

7.5.3 Comparison with other approaches

Table 7.4 shows the best results of our contributions together with the results of other works using the same experimental conditions (corpus and language modeling) reported in the literature. Careful attention must be paid when comparing different performances even if they use the IAM database. As mentioned above, we have used lines for training and evaluation. As can be observed, the proposed methods outperform all of the systems which recognize at line level. Nevertheless, other works used the whole paragraph for training and evaluation and included other features such as a variable number of states per character or writer adaptation Kozielski, Doetsch, et al. 2013. They reached a test WER of 13.3 for a vocabulary of 50K. Doetsch et al. Doetsch et al. 2014 achieved a WER of 12.2. Moreover, Bluche, Ney, and Kermorvant 2014 achieved a test WER of 11.9 with a vocabulary of 50K, which is the best performance obtained so far. This result was obtained with a ROVER system that was composed of the four possible combinations of HMM/ANN (deep MLPs) and BDLSTMs systems on the one hand and raw inputs and features extracted by specific parametrization process on the other hand. Pham et al. [2014] recognized, as well, at line level, but using a sort of n-best list in order to take full advantage of the language model, obtainig a WER of 13.6. Finally, the best results so far are achieved by [Poznanski et al. 2016] (a WER of 6.45 and a CER of 3.45), yet the word segmentation is assumed and the recognition is applied word by word. Besides, all train and test word occurrences are included in the system vocabulary.

7.6 Summary

In this work, we have presented several improvements to our recognition engine by removing feature extraction from the text images and using deep learning techniques directly on the text images. Deep MLPs and CNNs have been analyzed for the current task. The results presented in the IAM Database validate these approaches. In addition, we also studied the kind of features learned by the neurons by plotting some samples.

Although several CNN topologies are explored, one of the configurations that led to good results is comprised of a single convolution layer without pooling, achieving a WER of 17.1. If we compare this result with the baseline (HMM/ANN with features system which has a WER of 19.0), a considerable step forward in the recognition performance has been achieved. Further ex-

System	$\|V\|$	Test Set WER	Test Set CER
Isolated word recognition			
Bianne-Bernard et al. 2011	10K	32.7	-
Bluche, Ney, and Kermorvant 2013b	10K	20.5	-
Poznanski et al. 2016	No-OOV	6.29	3.37
Line recognition			
Bertolami and Bunke 2008	20K	32.8	-
Plötz et al. 2009	-	28.9	-
Graves, Liwicki, et al. 2009	20K	25.9	18.2
Toselli, Romero, et al. 2010	9K	25.8	-
Dreuw et al. 2011	50K	28.8	10.1
Espana-Boquera et al. 2011	20K	22.4	18.6
F. Zamora-Martínez, Frinken, et al. 2014	103K	20.0	8.3
F. Zamora-Martínez, Frinken, et al. 2014 (ROVER)	103K	16.1	7.6
Pastor-Pellicer et al. 2014	103K	24.4	10.6
Pastor-Pellicer et al. 2015 (baseline)	103K	19.0	7.5
Presented Approach (1)	103K	17.5	6.6
Presented Approach (2)	103K	17.1	6.3
Paragraph recognition			
Kozielski, Doetsch, et al. 2013	50K	13.3	5.1
Doetsch et al. 2014	50K	12.2	4.7
Bluche, Ney, and Kermorvant 2014	50K	11.9	4.9

Table 7.4: Performance for the IAM database. Approach 1 is using deep MLPs and raw input, and approach 2 is using the HMM/CNN system.

periments and the combination of different systems are expected to improve the results.

Chapter 8

F-Measure as Neural Network optimization function

Contents

Imbalanced datasets impose serious problems in Machine Learning. For many tasks characterized by imbalanced data, the F-Measure is commonly used when discussing the results and comparing them with other approaches. This chapter studies the use of F-Measure as the training criterion for ANN by integrating it in the Backpropagation algorithm. This novel training criterion has been empirically validated on the document cleaning and enhancing of documents.

8.1 Motivation for a new training criteria

It is not uncommon in many real tasks that the number of patterns of one class is significantly lower than other classes. Examples of tasks with very imbalanced data are information retrieval (a lot of information and very few useful data) or medical diagnosis (less ill than healthy patients). Imbalance datasets impose serious problems in machine learning and, particularly, in

Artificial Neural Networks (ANN) training. In those cases is common to use F-Measure (FM) when discussing the results and comparing them with other approaches.

Without going any further, we had an imbalanced task in Chapter 4, which is Document Image Binarization. Due to nature of the task, the FM is one of the most suitable measures for evaluation, indeed it is taken as the primary measure of the DIBCO evaluation.

Backpropagation algorithm is commonly used to train ANNs, but traditionally we fall in two error functions: MSE or CE error. This lead us to the following question: *Since the evaluated function is* FM *would it make more sense to use* FM *as the loss of our nets?*. In fact, training our method with FM woul also help with the unbalanced class problem. MSE and accuracy are good general measures, but it is not uncommon for many real tasks that the number of patterns of one class is significantly lower than other classes. Some authors have addressed this problem by resampling the data to balance the occurrences, others have modified the training algorithm [Al-Haddad et al. 2000; Z.-H. Zhou et al. 2006].

The motivation of this chapter is to design and test new training algorithm which uses the FM as an objective error function for the BP algorithm. In this chapter, we present the steps followed to applying the FM as a new target error function and the related experimental setup for DIB, yet it can be used in many other information retrieval tasks.

The main issue we had to deal with is that FM is a global optimization function, meaning that we can measure the FM on a set of retrieved values but not on an single is pixel. So when applying the new approach, we have to process a batch of several pixels to update the weights since it does make sense to compute the loss of only one pixel.

Though there are different approaches for the optimization of the FM using supervised techniques like SVMs [Musicant et al. 2003], logistic regression [Jansche 2005] and other ML techniques [Dembczynski et al. 2011], no such algorithm existed for ANN to the best of our knowledge.

We have used the FM training function for the DIB task, although it could be utilized for different training tasks which can be seen as information retrieval problem or even generalize the FM for more than one class and apply it as error training function when the datasets are unbalanced.

8.2 Error-backpropagation with F-Measure

Since the output of the ANN is represented as a real-value (i.e in the DIB task we use a logistic neuron as output), it is straightforward to compute a "soft" FM and error derivatives interpreting the output of the model as a probability, where the value for a pattern i is set as $o^{(i)} = P(\text{relevant}|\text{sample})$ and $1-o^{(i)} = P(\text{non-relevant}|\text{sample})$.

Following we present the formulation of the FM and the operations performed to derivate the error through the layers. For that, we must recall that FM is a quality measure computed as a combination between Precision (PR) and Recall (RC). For our task, it is possible to compute a version of the FM interpreting the output of the model as a binary value (for 2-class problems: 1 for relevant and 0 for non-relevant), being $o^{(i)}$ the output of the model for pattern i and $t^{(i)}$ the real-class value (0 or 1) for pattern i.

The computation of FM is a harmonic mean of PR and RC, and leads to the final formulation of FM in terms of *true positives* (TPs), *false positives* (FPs) and *false negatives* (FNs).

TPs, FPs, and FNs are computed over a dataset of m patterns:

$$\begin{aligned} TP &= \sum_{i=1}^{m} o^{(i)} \cdot t^{(i)}. \\ FP &= \sum_{i=1}^{m} o^{(i)} \cdot (1 - t^{(i)}). \\ FN &= \sum_{i=1}^{m} (1 - o^{(i)}) \cdot t^{(i)}. \end{aligned} \tag{8.1}$$

FM is formalized for positive real β, which weights the importance of recall versus precision, although the formula can be simplified by substituting TPs, FPs and FNs with previous definitions:

$$FM_\beta = \frac{(1+\beta^2)\cdot PR\cdot RC}{\beta^2\cdot PR + RC} = \frac{(1+\beta^2)\cdot TP}{(1+\beta^2)\cdot TP + \beta^2\cdot FN + FP} = \quad (8.2)$$

$$= \frac{(1+\beta^2)\cdot \sum_{i=1}^{m} o^{(i)}\cdot t^{(i)}}{\sum_{i=1}^{m}\left(o^{(i)} + \beta^2\cdot t^{(i)}\right)}. \quad (8.3)$$

to use the F-Measure as the objective error function in BP algorithm it is required to derive it by $o^{(i)}$:

$$\frac{\partial FM_\beta}{\partial o^{(i)}} = \frac{(1+\beta^2)t^{(i)}}{\sum_{j=1}^{m}\left(o^{(j)} + \beta^2\cdot t^{(j)}\right)} - \frac{(1+\beta^2)\cdot \sum_{j=1}^{m} o^{(j)}\cdot t^{(j)}}{\left[\sum_{j=1}^{m}\left(o^{(j)} + \beta^2\cdot t^{(j)}\right)\right]^2}. \quad (8.4)$$

Since BP is defined for minimization, the sign of the FM function has to be inverted. Note that the F-Measure derivative of pattern i depends on the others $m-1$ patterns, so it is not separable as the MSE or CE error. Therefore, the exact computation of this derivative forces to use batch training mode. However, batch training is slow and inaccurate when the number of patterns m is large (in the reported experiments, millions of samples). Because of these issues, we decided to use a mini-batch training mode, which leads to an approximation highly correlated with the actual FM computed on the entire dataset.

Another option is to split the image in patches and use these patches as batches since all the pixels of the patch are correlated. The main problem with this idea is that most of the patches will not contain foreground pixels which lead to a '0' weight update. We will see that is possible to take batches of random patterns and compute the FM errors on this randomly sampled batches. But still, the use of batch mode combined with random replacement makes it possible to sample a bunch of patterns where every target is *non-relevant*, meaning that these mini-batch presentations do not update the weights. This problem becomes more likely the lower the mini-batch size and also if the dataset is more imbalanced. Since each sample selection is independent of others, the probability of occurrence of this situation can be easily computed from the mini-batch size b and the proportion of 0's in the entire

training dataset (of size m) as $(F/m)^b$, where $F = \sum_{j=1}^{m}(1 - t^{(j)})$. This issue reduces convergence speed because mini-batch presentations suffering this problem do not update weights even if the output of the model is not correct.

8.3 Experimental Setup and Results

This section shows the experimental setup followed to verify our FM optimization function. In our experiments we have used the DIBCO dataset as well, presented in C.1. Nevertheless, there are slight differences from the DIBCO dataset used in the rest of DIB experiments (Section 4.7). In this case we did not used the DIBCO-2013 data, instead we took the following distribution:

- Training set: includes DIBCO-2009 and DIBCO-2010 datasets. A total of 24 images (19.8 Mpx, 6.6% classified as ink).
- Validation set: includes DIBCO 2011 dataset. A total of 12 images (10.0 Mpx patterns, 9.0% classified as ink).
- Test: includes DIBCO-2012 dataset. A total of 14 images (19.2 Mpx, 6.7% classified as ink).

The main reason for this partition arises from the fact that DIBCO-2013 was not released when we performed this experimentation. However, the sets taken sufficed for showing that our approach worked.

To evaluate and check the proposed technique, different configurations have been tried which differ in the error criteria:

- Logistic output unit ANNs trained using the MSE error criteria.
- Logistic output unit ANNs trained using the FM error criteria.

The training data have been used to find a common topology which works fine with both error criteria. Validation data was used to adjust parameters afterward. Finally, the trained networks have been used to compute the performance on the test set. Each type of error criteria has been tested on a network which shares the same input, hidden and output topologies. The input layer is composed of 90 input neurons: 81 pixels corresponding to a window of size 9×9 centered at the pixel to be cleaned and 9 additional context pixels associated with a 3×3 window with an estimation of background using a median

	Validation Data		Test Data	
	$\mu \pm \sigma$ MSE	$\mu \pm \sigma$ FM	$\mu \pm \sigma$ MSE	$\mu \pm \sigma$ FM
MSE train	0.0254 ± 0.0010	0.708 ± 0.013	0.0165 ± 0.0004	0.754 ± 0.007
FM train	0.0376 ± 0.0036	0.774 ± 0.012	0.0181 ± 0.0006	0.836 ± 0.009

Table 8.1: Average and Standard Deviation of theMSE and FM.

filter. [1] Regarding the hidden layers, the best configuration was two hidden layers of sizes 64 and 16, respectively. Also, 9 different randomly initialized networks have been trained to reduce the effect of local minima.

Table 8.1 shows the average of the FM and MSE measures, along with the standard deviation on validation and test sets for training for a mini-batch of 32 in both cases samples. Also, an example of a test set image cleaned with both ANN is depicted in Figure 8.1.

In general, both training techniques performed quite well when measured either on MSE or FM, since a well-cleaned image gives good results on both metrics. The results are not competitive compared with the best contest approaches [Pratikakis et al. 2010], although they are better than Method 1 which is also based on ANN (they obtained a FM of 0.82, and we got 0.836). We can also observe, from Table 8.1, that ANN trained with the FM error function obtain better FM than nets trained with the MSE error function. Conversely, the second model outperforms the first one in MSE in both validation and test sets. As expected, each training criteria prioritizes a different goal.

Next, to study the influence of the size of the mini-batch, different trainings have been carried out varying this parameter, and the reported results are illustrated in Figure 8.2. Two different factors may influence the results in opposite ways: on the one side, the larger the mini-batch size, the more accurate the approximation to the real FM should be. On the other hand, a smaller mini-batch size corresponds to a training scheme closer to the online version of BP which may have faster convergence. As it can be observed, as the mini-batch size is increased, the F-Measure performs worse, which means that even smaller mini-batch sizes may be highly correlated with the global FM.

Finally, to study the correlation between mini-batch size and FM, a statistic experiment has been carried out (see Figure 8.3) obtaining a Pearson product-

[1] Note that this is the net corresponding to the MLP with Features presented in Section 4.5.2. In this case, we did not use any histogram based features, only the median filter.

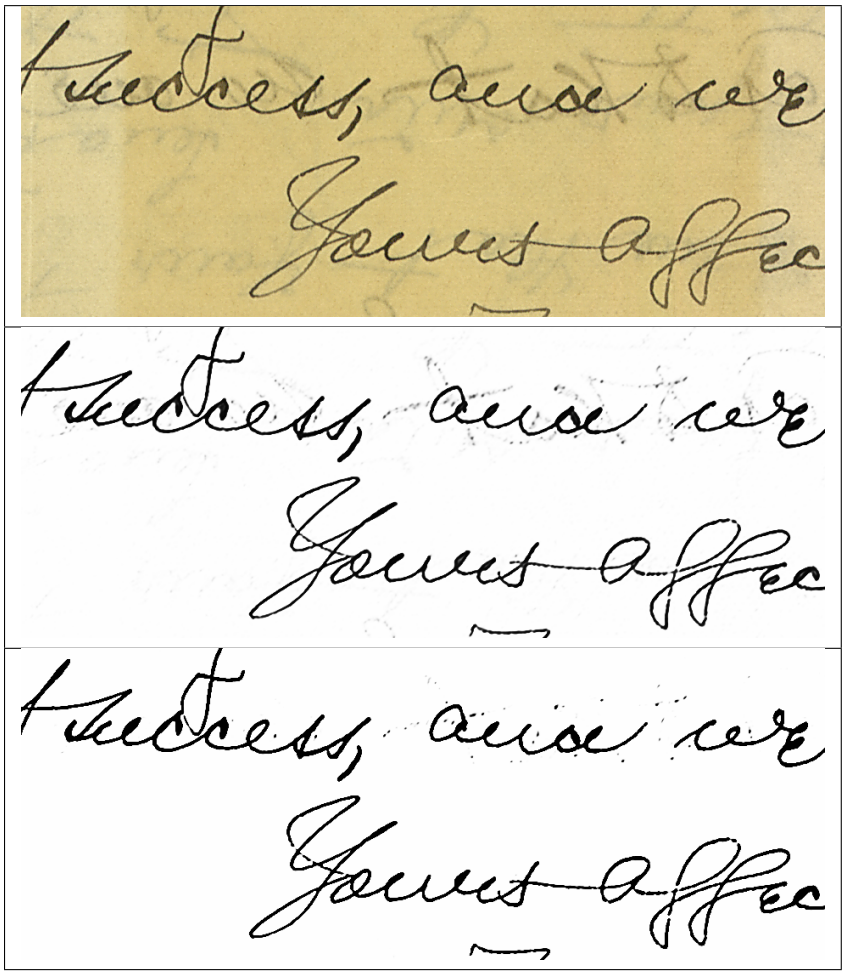


Figure 8.1: (Top) Example of a noisy test image. (Middle) The same image cleaned with the ANN trained with the MSE error criteria. (Bottom) The same image cleaned with the F-Measure error criteria.

moment correlation coefficient $r = 0.9991 \pm 0.0004$ with a confidence interval 99.9% ($p < 0.001$).

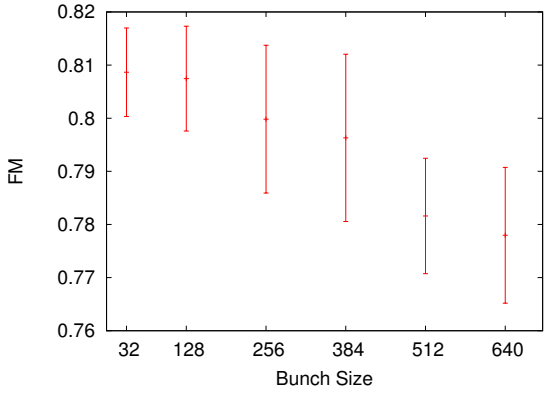


Figure 8.2: Influence of mini-batch size on the FM loss function.

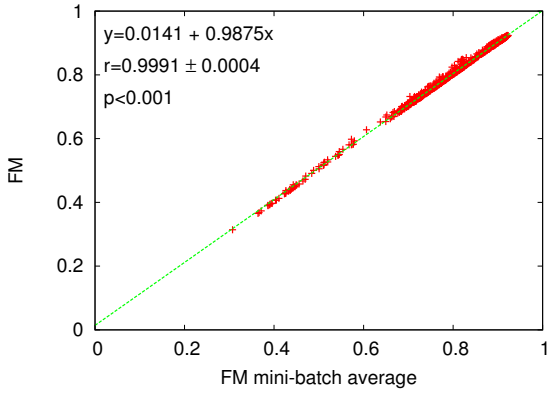


Figure 8.3: Correlation between the average FM of $100\,000$ mini-batch presentations of size 32 taken randomly and the FM value computed on the concatenation of all validation images.

8.4 Summary

We studied the use of more accurate and suitable error function such as the ingretion of the FM loss criterion in the BP process. Additionally, it has been explained how it can be adapted to batch training. We used the DIB task to empirically validate the new training criteria, motivated by working in a highly imbalanced task comprising millions of patterns (assuming each pixel is pattern). Experimental results show that, although FM trained with MSE or FM perform quite similar, each training mode prioritizes its corresponding assessment measure. This error criterion could be evaluated, as well, in tasks where FM makes sense, as is the case of information retrieval or document classification.

This work was collected in the following publication: [Pastor-Pellicer, F. Zamora-Martínez, et al. 2013].

Chapter 9

Conclusions and Future Work

Contents

In this chapter, we summarize the main discussions derived from this work. We also include an overview of the main contributions of our research. Finally, future lines of works and extensions are presented.

9.1 What has been accomplished

As introduced, one of the primary goals of this PhD Thesis was to increase the scope of our research, which is focused on text line HWR in order to include the main pre-processing stages. We think that our commitment has been fulfilled. On the one hand, document cleaning and enhancement has been applied at the page level. On the other hand, we have dealt with image segmentation tasks to extract the lines. With these new stages, we complete our recognition pipeline since it is feasible to obtain the transcription of a scanned image. First, the image is cleaned. Then, the lines are extracted, and, finally, the recognition is performed line by line.

With regard to the technologies developed and applied, one of the highlights of this work is the application of deep learning to HWR, avoiding specific fea-

ture extraction. To this end, CNNs and other deep ANNs, which have given promising results in Computer Vision tasks, have been used for this purpose. Following we summarize the work developed in this PhD Thesis.

One-to-one pixel-labeling Our contribution to one-to-one pixel labeling is introduced in Chapter 3. We implemented this approach by means of a sliding window to classify each pixel by CNNs, taking a surrounding area of that pixel as input. This was useful for DIB related tasks, but then, by adding more classes in the output, we labeled pixels in the first step for TLE and TLN tasks. One perk of this approach is that we could improve the pixel classification, thereby improving the overall performance of the system. Indeed, the technique is meant for HWR, but it could also be applied to other tasks such as image segmentation, text scene localization, and other pixel labeling-related tasks.

Document Image Binarization The improved one-to-one pixel labeling has mainly been applied for DIB and image cleaning and enhancement. We developed several methods based on ANNs to clean images. Our main contribution was to apply CNNs for this task. However, we also analyzed other techniques: MLPs (including extra features) and even MDLSTMs. For evaluation purposes, we also collected several corpora (Appendix C.1) and discussed the convenience of the several measures proposed for this task. We have demonstrated that supervised DIB methods are suitable for homogeneous collection even if they are noisy or present several distortions and degradations.

Text Line Extraction We introduced text line segmentation in our HWR pipeline. It is very rewarding to create a text line segmentation from scratch and get reasonably good results.

In the first approach, we collaborated with DIVA research group at Fribourg, and we combined our extracted segments with the final text line aggregation process. Their support and collaboration have been very valuable, and the results have been very promising. Finally, we again applied the one-to-one pixel labeling in the whole image to detect the Main Body Area of the text and then segment the lines. These approaches got very competitive results, especially in historical documents.

We realized that there is no a solid agreement about how to evaluate TLE-related tasks. The predicted and ground truth lines are coded as shapes (polygon, bounding boxes, pixel labeling), which have to be assigned/aligned and

then an error measurement must be computed. We included an interesting discussion about the evaluation metrics as well as our new proposals.

Text Line Normalization Extracting IPs for tracking the reference lines helped in the previously developed works for TLN. We continued this line of research and improved the text height normalization step. Our most significant contribution on this topic was to avoid the extraction of IPs by classifying all of the pixels of the image. Once again we were able apply our one-to-one pixel labeling by using CNNs to obtain a MBA map. After some pre-processing and using a DP algorithm inspired by the Seam Carving method, the text reference lines were obtained. This novel approach gave us an improvement in the recognition rate, improving our models by an absolute WER of 2 points (10.0% relative) and setting a new standard for normalization during the rest of the recognition experiments.

Decoding In this work, a lot of effort has been invested in improving the current group HWR engine. By identifying its deficencies and also analyzing the new improvements in HWR and deep learning, the overall line transcription process has been updated, and a significant improvement in the performance has been achieved. The original recognizer got a WER of 22.4, and, at the end of our developments, we reached 17.1 with the data, and LM [Espana-Boquera et al. 2011].

F-Measure as Artificial Neural Network optimization function A novel objective error function for the BP algorithm is proposed based on the FM. Additionally, it has been explained how it can be adapted to mini-batch training mode of BP. In order to empirically validate this training mode, a real task (DIB) using an imbalanced dataset of several millions of patterns has been carried out. Experimental results show that, although ANNs trained with MSE or with FM performs quite similar, each training mode prioritizes its corresponding assessment measure.

Other achievements We have also contributed by making publicly available the Noisy Office database described in Appendix C.1. The corpus consists of images of printed text documents with noise mainly caused by uncleanliness from a generic office such as coffee stains and footprints on documents and folded and wrinkled sheets with degraded printed text. This corpus is intended to train and evaluate supervised learning methods for cleaning, binarization and enhancement of noisy grayscale printed text image.

The described techniques required to improve and update our ANN framework: April-ANN. Almost all the ANN approaches followed in this PhD Thesis, specially CNNs, have been integrated into it. Besides, several functionalities and efficiency related improvements were carried out in the mentioned toolkit.

9.2 Half way

During the development of this PhD Thesis, there have been a lot of works and lines of research that were started but not been adequately explored or finished. Sometimes the effort required to achieve a result was not worth it. There were also situations where the first results were discouraging, so we quit those lines of research. However, the lack of time and need to prioritize other works and tasks wer the main reasons for not fulfilling the following duties. Nevertheless, some of the ideas/projects/techniques that we started are worth mentioning, and with enough effort and time, they would probably become interesting approaches. Indeed, some of these tasks are included in our future lines of work. We describe all these adventures, in the following paragraphs.

- **Apply the developed TLE models to other corpora.** We applied our text line segmentation approaches to other corpora, i.e., the *ICDAR Handwriting Segmentation Contest* [Hedjam, Nafchi, et al. 2015; Stamatopoulos et al. 2013]. We trained our models for the 2015 edition, and then we ran some preliminary evaluations on them. The biggest problem we encountered was that there were three different scripts: Latin, Greek, and Hebrew. Our models could be trained for Greek and Latin, but they were not suitable for Hebrew. Therefore, we had to change the set partitions, so our results were not comparable with the ones reported.

  With regard to historical documents, the Esposalles Database [Romero, Fornés, et al. 2013] was an excellent collection to try with our approaches.

Indeed, we applied the bootstrap supervising and cleaning methods presented in Chapter 4 to clean the pages of the corpus prior to TLE. We supervised the IP and MBA as explained in Appendix E. However, due to lack of time and other issues with the data, we decided to postpone this task.

- **Make our approaches available for external usage.** For TLE, we trained some models for the Historical IAM Database, and some preliminary models were obtained for other corpora. All of these models could be used, tuned, and adapted for other collections. We started making our algorithms available by means of web services [Marcel et al. 2016]. The primary challenge was to combine all of the steps in one process and allow our server to process the petition in time. For example, the MBA-based model presented in 5.6 has two main parts (MBA extraction and post-processing) which had to be adapted for this purpose.

  We also developed web demos based on web services for some of the neural filters presented in Chapter 4.

- **Distilling Neural Networks.** We tried to "distill" our networks by following the ideas introduced in [G. Hinton, Vinyals, et al. 2015]. We distilled the neural filters used in Chapter 4, but the first results did not seem very promising, so we decided not to continue this line of work.

- **Fully Convolutional Neural Networks.** As explained in 3.7, our approach differs from the known Fully Convolutional Neural Networks that are used for pixel labeling as well. In our approach, we had a more through process, but we obtained more features per pixel than the former one. We moved towards the use of Fully Convolution Neural Networks by classifying all of the pixels of the images jointly. Our first experiments used CNN without strides. Thus the original dimensions of the image are kept through the convolutions. We also explored a proper optimization with CUDA and GPUs for this matter, and we tackled the limitation of training with big images. Work along this line is still preliminary. Hence, we cannot show any relevant results or conclusions yet.

- **Foreground pixel labeling.** In the work developed for TLE and TLN, we started with IP extraction and classification, and then we moved to labelling all of the pixels of the image. Our original idea for height normalization was to classify only the foreground pixels of the image. We ran some initial experiments, but then we could not find any reliable way of splitting the image into zones with only the foreground information.

Classifying all of the pixels of the image led to MBA maps that were easier to segment.

- **Include our contributions in STATE.** In the research project HITITA (`http://blogs.uji.es/hitita/`), a transcription assistance tool (*STATE*) was developed by the team at *Universitat Jaume I*. Our recognizer was also included in this tool as a backend web service. Since the HWR recognizer was meant for normalized images, we rewrote the original IP normalization procedure to be used within the application; unfortunately, it was never included in the tool.

9.3 What should have been done

In the previous section, we listed things that have been partially done, and, unfortunately, most of them should have been finished. In this section, things that have not been explored totally or partially are listed. Our apologies to the reader for not being able to cover all of this research.

- **Apply the end-to-end to a full task.** We have extended our pre-processing steps to the full page process by including cleaning and line segmentation. With the new pre-processing stages, we deal with the necessary stages in order to have the complete stack of operations from the scanned image to the final transcription. However, unfortunately, in this PhD Thesis we could not apply the full process to any corpora since there is a gap between the data used for TLE and for the transcription. For TLE, we focused on historical documents while the recognition engine was set up for modern handwriting. On the one hand, it was not appropriate to apply our TLE extractions to the IAM Database since the input documents do not present a complex layout and TLE could be performed which easier techniques (e.g. histograms). On the other hand, we could have recognized the extracted lines in the historical documents. Once again, the reasons were due to scheduling; we decided to focus on the pre-processing stages and not to adapt the current HWR decoder to this data, which would have involved training new LMs and tuning some of the parameters of the recognizer.

- **Participate in DIBCO contests.** In Chapter 4, we used the DIBCO and H-DIBCO datasets as the main corpora evaluation. We could have submitted our approaches to these contests, but the submission required the preparation of a stand-alone executable file with some requirements.

Since it was not straightforward to adapt our toolkit and scripts to fulfill these demands, we decided not to invest effort in this matter.

- **Try many new regularization techniques that have been included in most of the toolkits.** During the elaboration process of this PhD Thesis, many ANN-related methods appeared and they were integrated into many toolkits. If we had presumed that these techniques could improve our classifiers, we would have considered them. However, there is a huge list of new techniques that we could have applied in our discriminative training. Implementing all of these techniques would have been tedious, but, indeed, most of them are integrated by the community/developers in open source toolkits like Theano, Torch, Tensorflow. This allows to apply them applied without significant coding effort. Here is a list of some of the trends in ANN training, that we should have, at least, given a try:

  - Batch Normalization/Layer Normalization [Ba et al. 2016; Ioffe et al. 2015]
  - Inception Nets [Szegedy et al. 2014]
  - Residual Networks [He et al. 2015]
  - DropConnect [Wan et al. 2013]
  - Elastic distortions [D. Cireşan et al. 2012]
  - Different weight initialization processes [Glorot et al. 2010]

9.4 Derived publications

Publication in ranked conferences (CORE)

- Pastor-Pellicer, J., Zamora-Martínez, F., España-Boquera, S., and Castro-Bleda, M. J. (2013). F-measure as the error function to train neural networks. In International Workshop on Artificial Neural Networks (IWANN) (pp. 376–384).

- Pastor-Pellicer, J., España-Boquera, S., Zamora-Martínez, F., and Castro-Bleda, M. J. (2014). Handwriting Normalization by Zone Estimation Using HMM/ANNs. In Frontiers in Handwriting Recognition (ICFHR) (pp. 633–638).

- Pastor-Pellicer, J., España-Boquera, S., Castro-Bleda, M. J., and Zamora-Martínez, F. (2015). A combined Convolutional Neural Network and Dynamic Programming approach for text line normalization. International

Conference on Document Analysis and Recognition (ICDAR) (pp. 341-345).

- Pastor-Pellicer, J., España-Boquera, S., Zamora-Martínez, F., Zeshan Afzal, M., and Castro-Bleda, M. J. (2015). Insights on the use of convolutional neural networks for document image binarization. In International Workshop on Artificial Neural Networks (IWANN) (Vol. 9095, pp. 115–126).
- Pastor-Pellicer, J., Castro-Bleda, M. J., and Adelantado-Torres, J. L. (2015). esCam: A Mobile Application to Capture and Enhance Text Images. In International Work-Conference on Artificial Neural Networks (pp. 601–604).
- Pastor-Pellicer, J., Afzal, M. Z., Liwicki, M., and Castro-Bleda, M. J. (2016). Complete System for Text Line Extraction Using Convolutional Neural Networks and Watershed Transform. In Proceedings - 12th IAPR International Workshop on Document Analysis Systems, DAS 2016 (pp. 30–35).

Non ranked conferences

- Pastor-Pellicer, J., Garz, A., Ingold, R., and Castro-Bleda, M.-J. (2015). Combining Learned Script Points and Combinatorial Optimization for Text Line Extraction. In Proceedings of the 3rd International Workshop on Historical Document Imaging and Processing (pp. 71–78).
- Afzal, M. Z., Pastor-Pellicer, J., Shafait, F., Breuel, T. M., Dengel, A., and Liwicki, M. (2015). Document Image Binarization using LSTM: A Sequence Learning Approach. Third International Workshop on Historical Document Imaging and Processing, (pp. 79–84).
- Adelantado-Torres, J. L., Pastor-Pellicer, J., and Castro-Bleda, M. J. (2014). Una aplicación móvil para la captura y mejora de imágenes de textos. V Jornadas TIMM.

Pending

- (Under revision) Castro-Bleda M.J., España-Boquera S., Pastor-Pellicer J., and Zamora-Martínez F.(2017). The NoisyOffice Database: A corpus to train supervised machine learning filters for image processing. International Journal on Document Analysis and Recognition (IJDAR).
- (Submitted) Pastor-Pellicer, J., Castro-Bleda, M. J., España-Boquera S., and Zamora-Martínez, F. (2017). Handwriting recognition by using deep learning techniques to extract meaningful features. AI Communications.

Other publications

- Zamora-Martínez, F., España-Boquera, S., Gorbe-Moya, J., Pastor-Pellicer, J., and Palacios-Corella, A. (2013). APRIL-ANN toolkit, A Pattern Recognizer In Lua with Artificial Neural Networks (`https://github.com/april-org/`).

9.5 Future work

As future work we plan to continue our research by implementing and finishing the ideas mentioned in the above sections. In addition to these, we have established the following new lines of research and techniques to improve the methods presented in this document.

In Chapter 4, we tried different ANNs and we combined them for cleaning and enhancing document images. It is evident that newer models could be added to this task, which certainly will outperform the given approaches. For example, applying a net that is similar to [D. C. Cireşan et al. 2012] which uses multi-column deep CNNs that allow using different resolutions of the image simultaneously. Another line of research is to use Fully Convolutional ANNs, especially deconvolutions, and increasing the number of maps in each convolution. We can try more ideas on this topic such as the use of 2D convolutions and MDLSTMs, which work as convolutions but keep an internal state. Therefore, it is straightforward to stack CNNs and LSTM. Particular attention must be established for the training in order to avoid gradient explosion or even to have a memory-efficient algorithm due to recurrence connections. It is also straightforward to adapt these pixel-labeling ideas to MBA detection-based tasks: TLE and TLN. However, in these cases, other post-processing techniques are applied after the ANN classification: hence the impact of these techniques will be smaller than in a pure pixel -labeling task.

With regard to TLE, we presented two main approaches. With respect to the Combination Optimization Problem (Section 5.5.1), we plan to apply it to more complex documents like embellished manuscripts. The MBA-based approach could be improved by a better detection of the ascenders and descenders. Remember that it detects the central zone of the line and the last contours are extracted later. We plan to improve this stage by having a more reliable frontiers once the line has been detected.

On height normalization, the new techniques avoid the classification of IPs, but both approaches (MBA estimation and IP-based [Gorbe-Moya et al. 2008]) could be combined in order to overcome their weakness and to obtain a more robust system. We plan to use these approaches in other pre-processing stages; for instance, it is straightforward to adapt both techniques for slope and skew correction. For each image column, we just have to move the MBA to the center of the image. For slant correction, some further work is required to adapt the current models, but the final purpose is to fit all of these pre-processing steps into only one process: scan the image using an ANN and then apply all the full normalization on the generated maps. We also plan to evaluate the system on other corpora to check the robustness of the proposed approach.

With regard to HWR and decoding, the baseline recognizer has been improved by avoiding specific feature extraction and by applying deep learning in the optical modeling. Before trying new improvements, we plan to do proper training with paragraphs instead of lines as in [Bluche, Ney, and Kermorvant 2014; Kozielski, Doetsch, et al. 2013]. We then propose to try deeper architectures such as Residual Nets and also to apply other normalization techniques such as batch/layer normalization to speed up the training in order to obtain better results. Finally, we need to do a more thorough error analysis to determine which steps of the whole transcription pipeline we should focus on to assure new improvements.

In conclusion, we would like to try newer and more complex nets: ResNets, Generative Adversarial Networks. For this purpose, we intend to optimize our techniques in order to apply them in a reasonable amount of computation time.

Appendices

Appendix A

Description of the APRIL-ANN toolkit

During our research at The Natural Language Engineering and Pattern Recognition Group (ELiRF) research team we started developing a toolkit for ANN and other pattern recognition algorithms. The development started at 2005 and continues today. The first idea of this project was to develop a suitable infrastructure for fast development and high performance of pattern recognition algorithms. The APRIL-ANN toolkit combines fast operation algorithms developed in C++ and Lua for scripting [F. Zamora-Martínez, España-Boquera, et al. 2013]. It uses its own binding definition between Lua objects and C++ classes. Several algorithms for ANNs and HMM as well as Image Processing were coded and extensively used in our research. Following, great improvements came to APRIL-ANN, especially about ANN and matrix operations. After that, the code was released under GPL license and published on GitHub by the name of *APRIL-ANN* (A Pattern Recognition in Lua): `https://github.com/pakozm/april-ann`. The developers of our group are still improving and extending the functionalities.

APRIL-ANN uses matrix-based objects for ANNs computations. The code can be compiled with different mathematical libraries for fast matrix computation. In addition, algebraic operations and memory blocks are implemented by using wrappers for performing the computation in CPU and/or GPUs. The basic (slower) compilation uses ATLAS for mathematical matrix operations. Intel MKL is recommended for Intel processors, and finally, CUDA compilation uses

cuBLAS among other CUDA libraries for the basic toolkit operations. High-level objects in Lua are transparent for the implementation used, which makes it easier the development of algorithms without relying on architecture issues. Under the hoods, algorithms are optimized to work with the desired library.

In addition to these performance features, APRIL-ANN provides a component-based structure for ANN algorithms, like Torch and other toolkits do. Each component implements a typical ANN interface: forward, backprop and gradient computation. The final configured ANN iterates over the components graph which is the definition of the full neural network system. Following these ideas, an efficient and complete toolkit has been developed which provides several features and utilities for ANNs. You can made use of deep learning which is the case of the SDAE for layer-wise pretraining and CNNs among others. Recurrence is also provided and we are working on extending LSTM models to Bidirectional LSTM and Multidireccional LSTM. Also, several regularization and training related characteristics are included in the APRIL-ANN toolkit: weight decay regularization, sparse activation, dropout, adaptive learning rate methods, automatic differentiation and others.

Appendix B

Hybrid HMM/ANN Modeling

Chapter 6 and 7 showed approaches that rely on the use of full Handwriting Text Recognition (HWR) recognizer. The recognition engine used for this purpose was the one developed by the ELiRF team [Espana-Boquera et al. 2011; Gorbe-Moya et al. 2008] and it is based in HMM/ANN. The models developed for TLN used the original decoder, while in Chapter 7 we modified the Optical modeling part to avoid feature extraction.

In this chapter, we illustrate the Hidden Markov Models hybridized with ANNs (HMM/ANN) models applied for HWR which have been used during this PhD Thesis.

B.1 Handwriting Text Recognition (HWR)

Offline HWR could be seen as a left-to-right sequence of ink strokes. HWR engines receive a text line as input, generally the image is converted to a sequence $X = (x_1 \dots x_m)$ of feature vectors. The main goal is to find the likeliest word sequence ($W^\star = (w_1 \dots w_n)$) that maximizes the posterior probability:

$$W^\star = \underset{W \in \Omega^+}{\arg\max} \quad P(W|X) \ . \tag{B.1}$$

From all the possible sequences of words of a given vocabulary Ω, we want to know the one that has the maximum probability given the input set of features

$P(W|X)$. This formula is decomposed as the product of the optical model $P(X|W)$ and the statistical LM using the Bayes' theorem:

$$W^{\star} = \arg\max_{W \in \Omega^{+}} \quad P(X|W)P(W) \ . \tag{B.2}$$

On the one hand, for the current work the optical modeling $P(X|W)$ is estimated by a HMM over the sequence of features. The language modeling, as we detail in the following section, has been approached by the use of n-gram and connectionist LMs.

B.2 Optical models by Hybrid HMM/ANN

For the input features sequences, we want to find the sequence of words that provides the maximum likelihood $P(W|X)$. One way to achieve this is to model each grapheme as a left-to-right HMM, then for a given sequence of graphemes $W \in \Omega$ (word sequences are finally grapheme sequences with the special blank character), the $P(W|X)$ could be computed as the concatenation of each grapheme HMM in the sequence [L. R. Rabiner 1989]. The decoding stage with additional lexicon information will be lead the (beam) search of the possible $W \in \Omega$ sequences.

In Chapter 2.4, we reviewed related works based on HMM/ANN for HWR. As already state there, the HMM is defined by transition ($p(q_j|q_i)$ from state q_i to q_j) and emission probabilities ($p(x_n|q_i)$ for the frame x_n and the state q_i).

The emission probability density function ($P(x|q)$) is estimated for each state q, that is the probability of the observed feature vector x given the hypothesized state q of the model. It is possible to estimate the emission probability by a discriminative model such as MLP that approximates the a-posterior probabilities of the each state $P(q|x)$. The output of the network corresponds to the number of possible states in the HMM: given a set of n characters and s hidden states for each character, the output of the net will have $|n \times s|$ softmax output units, since they estimate probabilities [Bishop 1995; Bourlard et al. 1994].

By the Bayes theorem, the emission probabilities are obtained from the posteriors probabilities as:

$$P(x|q) = \frac{P(q|x)P(x)}{P(q)} \ . \tag{B.3}$$

We want to find the best alignment. Therefore, we can ignore the $P(x)$ since it is constant for all the possible alignment. It lead us to the *scaled likelihoods*:

$$P(x|q) \propto \ P(x|q) = \frac{P(q|x)}{P(q)^{\alpha}} \ . \tag{B.4}$$

The class priors $P(q)$ can be estimated from the relative frequencies of each state on the training set. For example, we could apply forced alignment to get an initial estimation. The α parameter weights the state priors ($0.0 \leq \ \alpha \leq \ 1.0$).

On the counterpart, the ANN receives a contextual field on the centered frame x. The approach runs like a sliding window of n frame neighbors, and the net receives $(n + 1 + n) * features$ input units.

B.3 Language Modeling

The second part of the equation corresponding to the $P(W)$ describes the a-priori probability of the sentence. Since the sentence is formed by a sequence of m words ($w_1, w_2, \cdots, w_m$), the probability could decomposed by the chain rule:

$$P(w_1, w_2, \cdots, w_m) = \prod_{i=1,\ldots,m} P(w_i|w1 \cdots \ w_{i-1}) \ . \tag{B.5}$$

From left-to-right each word is estimated as the previous history words, n-grams restrict the size of the history to $n - 1$ words:

$$P(w_i|w_{1\ldots \ w_{i-1}}) \sim P(w_i|w_{i-(n-1)} \cdots \ w_{i-1}) \ . \tag{B.6}$$

The word LM is integrated into the recognition process using the *word insert penalty* and *grammar scale factor* commonly applied in the domain of speech recognition.

Count based n-grams The estimation of the parameters of n-gram models is traditionally done by counting. The probability of a given n-gram is formulated as the counting of the n-gram by the occurrences of the $(n-1)$-gram:

$$P(w_i|w_{i-(n-1)}\dots\ w_{i-1}) = \frac{C(w_{i-(n-1)}\dots\ w_i)}{C(w_{i-(n-1)}\dots\ w_{i-1})}\ . \tag{B.7}$$

Since the data is sparse, it is required to use smoothness techniques. The main ideas are based on adding part of the probability mass to unobserved events or combine with back-off models.

Neural Network Language Models (NNLMs) NNLMs could estimate posterior probabilities such as the n-gram model:

$$P(w_i|w_{i-(n-1)}\dots\ w_{i-1})\ . \tag{B.8}$$

A basic setup is to take as input the $n-1$ words, and the output is composed by $|\Omega|$ softmax units, where each output unit j estimates [Bengio, Ducharme, et al. 2003; Schwenk 2007]:

$$P(w^j|w_{i-(n-1)}\dots\ w_{i-1}), w^j \in \Omega \tag{B.9}$$

Usually, input words are coded at the input as *1-of-k (one-hot-encoding)* of the size of size $|\Omega|$. Then, as seen in Figure B.1 a shared weights projection layer reduce the dimensionality of the sparse input.

B.4 HMM/ANN engine description (Baseline)

Once introduced the basics of our recognition engine, we concrete some of the parameters used. This architecture is fixed in this PhD Thesis recognition experiments unless we explicit mark the changes.

In our baseline system [Espana-Boquera et al. 2011], the emission probabilities are estimated by an MLP, whose input is a sequence of feature vectors following [Toselli, Juan, et al. 2004].

The recognition engine is based on a hybridized HMM with an MLP to model graphemes, which was presented in [Espana-Boquera et al. 2011]. Each grapheme

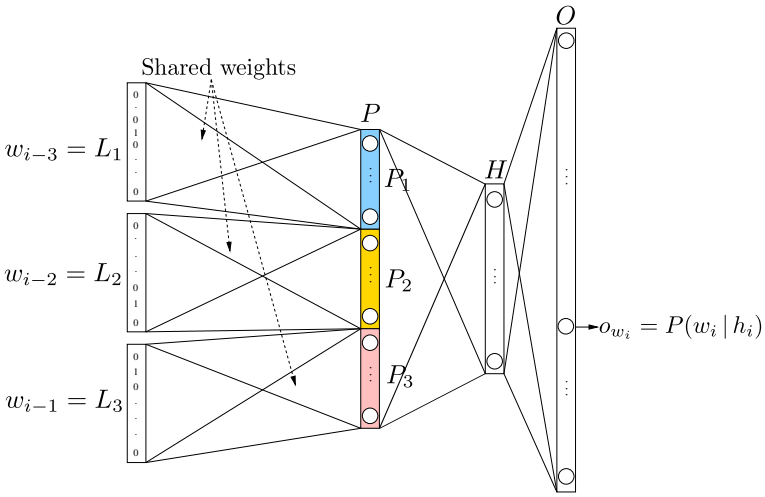


Figure B.1: NNLM used. It depicts a 4-gram. L indicates the *one-of-k* vectors, P the projection layer, H the note the hidden layer(s), and O the final output layer. [Figure extracted from [F. Zamora-Martínez 2012]]

is modeled with a 7-state left-to-right HMM topology with loops and without skips. The connectionist model used to estimate the emission probabilities of the HMM states was an MLP with 2 hidden layers of 512 and 256 units, respectively, using the softmax activation at the output layer. The HMM/ANN system is trained by means of an EM procedure with a forced Viterbi alignment.

The images received by the recognition engine were preprocessed following the skew and slant correction presented in [Espana-Boquera et al. 2011] and following the height normalization proposed in [Pastor-Pellicer, España-Boquera, F. Zamora-Martínez, et al. 2015]. For each HMM step, the parametrization obtained 11 contextual frames, which were extracted from a window of 28×42. W.r.t to the HMM, for each grapheme a 7-state left-to-right topology with loops and without skips has been chosen. On the traditional setup, a neural model for generating the emission probabilities of the HMM graphemes was trained by an MLP with 2 hidden layers of 512 and 256 units using the softmax activation at the output layer. One output unit for each state of every HMM was needed, for a total of 553 output units (79 grapheme models composed of 7 states each).

B.4.1 N-gram model

The n-grams language models were from [F. Zamora-Martínez, Frinken, et al. 2014] whose vocabulary size differs from the original work [Espana-Boquera et al. 2011]. A 4-gram with Witten-Bell smoothing that was trained with the SRILM toolkit [Stolcke 2002] was used. The text corpora used to train the n-gram LM were: the LOB corpus [Johansson et al. 1986] (excluding those sentences that contain lines from the test set or the validation set of the IAM task), the Brown corpus [Francis et al. 1979], and the Wellington corpus [Bauer 1993]. The lexicon of the LM had approximately 103K different words. Word insertion penalty and grammar scale factor parameters were optimized on the validation set by means of the MERT procedure [Och 2003].

Appendix C

Document Image Corpora

In this Appendix we collect the corpora used during our evaluation. We will include references, and used data distribution and partitions.

C.1 Document Image Binarization

In Chapter 4 we briefly disscussed different techniques to generate supervised corpora for the DIB task. As it has been stated, it is important to judge and evaluate our methods but also know the kind of data evaluated (distribution, the level of noise, type of documents, difficulty).

C.1.1 DIBCO

The Document Image Binarization Contest (DIBCO) [B. Gatos et al. 2009; Pratikakis et al. 2013] and the Handwritten Document Image Binarization Contest (H-DIBCO) [Pratikakis et al. 2012, 2010] are contests that have been hold in the context of the International Conference on Frontiers (ICFHR) in Handwriting Conference and International Conference on Document Analysis (ICDAR) and Recognition conferences since 2009.

In each edition, a new set of images has been provided for evaluation. Images from previous editions could be employed to train or verify the developed approaches. From now on, we are going to refer as DIBCO corpus as the set of supervised images of the several editions of DIBCO and H-DIBCO contests.

Using this corpus for evaluating has several advantages:

- It is a public competition, the images have been generated using novel noising techniques and they have been used and tested by different groups and companies.
- The ground truth is not expected to be ambiguous or subjective.
- It is easy to compare with the state-of-the-art methods since the evaluation and results are provided in each edition.

This dataset also has some drawbacks:

- Even tough each year new set of images is provided, the set of images of each edition is relatively small.
- The corpus has very different types of images, noises, fonts, and scales. It is hard to find two different pages with the same style, font size or background and, also, the sizes of the images differ.

These pecularities make that supervised based methods perform worse than other heuristics and adaptive methods. Even though, it is very interesting to see which one is performing better in this kind of conditions.

In past edition, supervised-based methods were trained using data of the contests and, additional training data. Hence it is common to use enriched corpora with more synthetic data. Another important point for success in supervised techniques is the capability of generalization when trained with heterogeneous data.

In our case, only data of previous editions has been used for training the ANN-based methods. Three partitions were used as shown in table C.1.

C.1.2 Saint Gall

The IAM Historical Saint Gall dataset has been used as well for our DIB evaluation purposes. More details of the IAM Historical Document Database (IAM-HistDB), are provided in the next section.

This document database is appropriate for supervised methods since the font and size of the text are homogeneous along the full collection, and there are

	Train		Validation		Test	
CORPUS	HW	TW	HW	TW	HW	TW
DIBCO 2009	7	7				
H-DIBCO 2010	10					
DIBCO 2011	8	5		3		
H-DIBCO 2012			12			
DIBCO13					8	8
Subtotal	25	12	12	3	8	8
Total	37		15		16	

Table C.1: Set distributions for DIBCO and H-DIBCO. There are two types of images: Handwritten (HW) and Typewritten (TW).

lots of pages and enough meaningful data. Besides, these documents are ideal to see how the evaluated method can learn text shapes and edges of the expected text. However, it is also convenient for adaptive methods with small tuning of the parameters they perform well for the rest of collection.

The dataset includes transcriptions and line descriptions. WER and CER would be good indicators about how our approaches improve the pre-processing but other DIA metrics could be evaluated.

Cropping Saint Gall images have been taking by a digital scanner, so there are black bands (out-of-page) on the extremes of the images. This is very common when the source of the data came from scanners and other specialized cameras. In this case, there is some ambiguity about how to deal with this:

1. Since there is no text and it is of the background, they are considered as background. That is class 0, or white pixels on the background.
2. Keep them as a foreground since there are black pixels they can be considered as black pixel or 1 label foreground pixels.
3. It is possible to remove this ambiguity applying basic heuristics.

Some supervised approaches can deal with this artifact due to they can learn to remove it, while other heuristics methods like Sauvola's and Otsu's will fail in this case. Anyhow, it is not a big problem because it is easy to remove this black background like where a simple heuristic is applied, and the images have been cropped to the sheet size.

The black bands constitute an important part of the image ($10/30\%$ of the pixels). So for our purposes we have generated a new dataset (Saint Gall - Crop) without the bands, we applied a separate vertical, and horizontal histogram for each row/column and a band is removed if the percentage of black pixels is above 90%.

For training and evaluation we have taken the sets depicted in the corpus description:

- Training: 20 pages corresponding to manuscript *Codex Sangallensis 562*, pages 3 to 20.
- Validation: 10 pages of manuscript *Codex Sangallensis 562*, pages 24, 28, 32, 36, 49, 44, 48, 55, 59 and 63.
- Validation: 30 pages of manuscript *Codex Sangallensis 562*, pages 23, $25-27$,$29-31$, $33-35$, $37-39$, $41-43$, $45-47$, 49, 50, 54, $56-58$, $60-62$, 64 and 65.

C.1.3 Noisy Office

This corpus comprises different types of noise added to a synthetic typewritten data. In this case, we have a document text templates which are faded with some background taken from real paper pictures.

Noisy Office can be considered a toy corpus since the resolution of the images is very low (small patches around $540x420$ pixels with $200ppi$. The foreground text is printed which no distortion. Moreover, the images have the same size and text style. This dataset is very suitable for machine learning methods.

On the other hand, it represents real and expected noise in an office environment: folded sheets, wrinkled sheets, coffee stains, footprints. The font is sans serif or roman with sizes: footnote, regular or large.

Besides, it presents some stains and other hard noise which is not easy to remove. Additionally, the computing of the ground truth is straightforward, due to the background and the noise have been added later to the synthetic data.

C.1.4 Corpora Information

Table C.2 shows some information about the corpora; these indicator try to illustrate the complexity of each corpus. It is hard to measure the quantity and level of noise in each since several kinds of noise could be appeared.

So we measure the size of the images which is directly proportional to the computation cost since each neural network is applied at pixel level. We also measure the signal/noise ratio with the Peak Signal to Noise Ratio ($PSNR$) which is computed as $10 \cdot log_{10}(\frac{1}{\sqrt{(MSE)}})$, where the MSE is the mean squared error between the dirty image and the ground truth, the lower this score, the noisier the outcome.

C.2 IAM Historical Document Database (IAM-HistDB)

The IAM Historical Document Dabase (IAM-HistDB) has been also used for evaluating the approaches developed in Chapter 4 and 5[2].

According to their web page:

> The IAM-HistDB is a repository of data sets that contain handwritten historical manuscript images together with ground truth data for training and testing automatic handwriting recognition systems
>
> The IAM-HistDB is compiled by three datasets: *Saint Gall*, *Parzival* and the *Washington* databases. We have mainly worked in the former two.
>
> The database was generated by the *Research Group on Computer Vision and Artificial Intelligence* at the University of Bern.

[2] http://www.fki.inf.unibe.ch/databases/iam-historical-document-database

DIBCO	Train	Validation	Test
Images	37	17	16
Total Pixels	28, 71Mp	20, 28Mp	29, 39Mp
Largest Image	2044×1308	2245×1317	4161×1049
Foreground pixels	0.09Mp (7.3%)	1.4Mp (6.9%)	1.98Mp (6.74%)
PSNR[1]	10.98	11.25%	10.88%

Saint Gall	Train	Validation	Test
Images	20	10	30
Total Pixels	332.26Mp	166.133Mp	498.401Mp
Largest Image	3328×4992	3328×4992	3328×4992
Foreground pixels	11.74Mp (3.53%)	5.34Mp (3.21%)	16.31Mp (3.27%)
PSNR	5.13%	5.13%	5.17%

Saint Gall (CROP)	Train	Validation	Test
Images	20	10	30
Total Pixels	252Mp	125.9Mp	379.03Mp
Largest Image	3328×3970	3195×3996	3230×3967
Foreground pixels	11.74Mp (4.65%)	5.34Mp (4.24%)	16.31Mp (4.3)
PSNR	10.35	10.52	10.54

OFFICE	Train	Validation	Test
Images	72	72	72
Total Pixels	142.3Mp	142.3Mp	142.3Mp
Largest Image	540×420	540×420	540×420
Foreground pixels	1.51Mp (10.63)	1.52 (10.62%)	1.44 (10.18%)
PSNR	13.55	13.78	12.34

Table C.2: Statistics about the binarization corpora.

	pages	**lines**	**words**	**w. labels**	**letters**
Saint Gall	60	1410	11597	4890	49
Parzival	47	4477	23478	4934	

	Training		Validation		Test	
	Pages	Lines	Pages	Lines	Pages	Lines
Saint Gall	20	471	10	1405	30	721
Parzival	24	2285	9	720	14	1405

Table C.3: Distribution of the Historical IAM-DB.

C.2.1 Saint Gall

The images of the Historical IAM Saint Gall Database are captured from the Manuscript images of the Codex Sangallensis 562. The dataset and contains a handwritten historical manuscript:

- 9th century.
- Latin Language.
- single writer
- Carolingian script
- ink on parchment

Distorted Saint Gall A distorted version of the Saint Gall dataset was created by [Kieu, Visani, Journet, Domenger, et al. 2012], we used it to asses our approaches as well.

C.2.2 Parzival

The PARZIVAL Dataset database is compiled from 13th century Gothic scripts:

- 13th century.
- Medieval German language.
- three writers.
- Gothic script.
- ink on parchment.

Table C.3 show some statistics of both corpora.

C.3 IAM Offline Database

The IAM offline dataset [Marti et al. 2002a] is composed of forms containing handwritten English sentences extracted from the LOB corpus [Johansson et al. 1986]. The version 3.0 of the IAM Dataset (`http://www.iam.unibe.ch/fki/databases/iam-handwriting-database`) has been mainly used for evaluating our developments in the recognition engine and text line pre-processings. This version collects 5685 sentences from 657 different writers, with a total of 115000 word instances, composed by a total of 78 different graphemes. The forms are divided into lines, which are the input for the experimentation of this work. The standard training and test partitions have been used.

C.3.1 Interest Points, Text Reference Lines and MBA ground truth

The reference lines from a subset of 773 text line images from the training data of the IAM database have been manually labeled using a bootstrapping approach and interactive tools, as described and used in [Espana-Boquera et al. 2011; Pastor-Pellicer, España-Boquera, P. Zamora-Martínez, et al. 2014; F. Zamora-Martínez, Frinken, et al. 2014]. This subset of lines has been split into training and validation.

Appendix D

Text Line evaluation and ground truth formats

D.1 Evaluation and metrics

Likewise with DIB (Chapter 4), it is possible to evaluate the performance of TLE directly on the text line detection or its impact in further stages, for example, the final transcription error (CER/WER). In this PhD Thesis we discuss and utilize text-line-level error metrics even though they could be misleading. For example, if we missed a part of a text line it could have a notable impact on the TLE metrics, but maybe, the lost stroke has not any effect on the final recognition. The opposite situation could happen as well: only a small part of the line is missed, but it has a bigger impact on the recognition rates. However, as always, we could assume some homogeneity and error correlation between the text line extraction performance and the following stages. It is reasonable to think that a wrong text line segmentation will lead to bad transcription results.

At line level, usually, the shared regions between ground truth and predicted lines are evaluated. Since the line frontiers could be ambiguous because of the white spaces between them, during the evaluation, only foreground pixels are taken into account. Two main drawbacks are drawn:

- We need a strong foreground definition. Therefore we have the issues related to the binarization ground truth described in Chapter 4.
- *How to treat grayscale images, should we weight the results according to the brightness valu of each pixel?*

The evaluation process compares a set of ground truth lines with the predicted lines. This could be approached as a precision/recall problem, such as an information retrieval task. In addition, the evaluation could be computed at two different levels:

- **Line level.** We count the line hits and misses according to matching criteria between predicted and ground truth lines. It estimates how many lines have been detected correctly according to some tolerance threshold.
- **Pixel level.** We could compute the error without relying on "all of nothing" line thresholding evaluation. The evaluation is given by counting the number of well detected pixels. Thus the ratio of foreground pixels that have been tagged correctly is estimated.

Match Score from ICDAR The Match Score (MS) evaluation has been applied in several text line segmentation contests: ICDAR 2009 Handwritten Segmentation Contest, ICFHR 2011 Handwritten Segmentation Contest and the ICDAR 2013 Handwritten Segmentation Contest [B. Gatos et al. 2009; B. Gatos, Stamatopoulos, et al. 2010; Stamatopoulos et al. 2013]. Besides, the proposed metrics in these contests have been widely applied for evaluation and comparison between text line extraction algorithms. This assessment has been adapted from [Phillips et al. 1999] which was initially proposed for graphic recognition systems.

MS counts the text line matches between the prediction and ground truth lines. Lets us have a set of N predicted and M ground truth lines. Let G be the set of M entities in the ground truth, and R the N objects predicted. It is a function that computes the intersection of the foreground pixels between a predicted line (R_i) and the ground truth line (G_j). For each pair of lines, it returns a real value $[0, 1]$ computed as:

$$MS = \frac{T(G_j \cap R_i \cap I)}{T(G_j \cup R_i) \cap I} \ . \tag{D.1}$$

pred \ gt	5	6	7
1	.79	.11	0
2	0	.68	.08
3	0	0	.86

Table D.1: Sample of the MatchScores between predicted lines $(1, 2, 3)$ and the gt lines $(5, 6, 7)$. For example the $MatchScore(1, 5) = T(G_1 \cap R_i)/T(G_1 \cup R_5) = 11/4 = 0.79$.

A $M \times N$ score matrix with all possible pairs is generated. Table D.1 has been taken from the illustrative example used on the evaluation procedure explanation[1].

After computing the MS matrix, we consider a *one-to-one* match ($o2o$) if a cell score is higher than certain threshold T_α (0.95 for handwritten and modern documents, and 0.9 for historical documents). Finally, the evaluation is expressed regarding precision and recall. Thus it assesses the trade-off between the Detection Rate (DR) and the Recognition Accuracy (RA). DR tracks the number of predicted lines that have been correctly assigned (precision), while the RA evaluates the correctly detected ground truth lines (D.2). Besides the F-Measure (FM) proposed computes the harmonic mean between the DR and RA.

$$DR = \frac{o2o}{N}, \;\; DA = \frac{o2o}{M}, \;\; FM = \frac{2DR\ RA}{DR + RA} \;\; . \tag{D.2}$$

Pixel Level Hit Rate (PHR) and Text-Line-Level Detection Rate (TLL-DR) The Pixel Level Hit Rate (PHR) and Text-Line-Level Detection Rate (TLL-DR) are the metrics proposed by [Y. Li et al. 2008]. These measures work either at the line and pixel levels. The evaluation at page level is similar to the MS since it counts the hits and misses of predicted/ground truth lines.

For computing both metrics, first, similar procedure than the MS function is followed. Generating, a $M \times N$ matrix with the scores of all the predicted/ground truth pairs. The score function ($P_{i,j}$) computes the shared foreground pixels between a predicted (Ri) and a ground truth line (Gj). Note that the score matrix, as we will see, differs slightly with the MS matrix computed previously:

$$P_{i,j} = T(G_j \cap R_i \cap I) \;\; . \tag{D.3}$$

[1] http://users.iit.demokritos.gr/~nstam/ICDAR2013HandSegmCont/Evaluation.html

Once the matrix is computed, both approaches diverge. In this case, the primary goal is to get one-to-one correspondence between each of the ground truth and predicted lines. If the number of lines differs, dummy lines are added. So we have a P' matrix of dimension $max(M, N) \times max(M, N)$. For each possible line assignment, the Goodness S is computed as the sum of all shared foreground pixels between the pairs of lines in the assignment.

$$Goodness(S) = \sum_{k=0}^{max(M,N)} P_{k,S(k)} \ . \tag{D.4}$$

Following, the best assignment is computed as the one that maximize the *Goodness*:

$$S_o = \arg\max_S \ Goodness(S) \ . \tag{D.5}$$

Finding the best assignment is solved applying combinatorial optimization algorithms. The authors used the Hungarian algorithm for an efficient computation in polynomial time. Once the best assignment is fixed, the PHR is defined as the number of shared black pixels between each of the pairs, normalized by the number of black pixels in the ground truth:

$$H = \frac{Goodness(S_o)}{\#\text{foreground}} \ . \tag{D.6}$$

Finally, it is computed the TLL-DR as the number of well-detected lines. A line is claimed to be detected if it shares with the predicted 90% of the foreground pixels:

$$\frac{P_{i,S_o(i)}}{\sum_{j=1}^{N} P_{i,j}} \geq 0.9, \ \frac{P_{k,S_o(i)}}{\sum_{k=1}^{N} P_{k,S_o(i)}} \geq 0.9 \ . \tag{D.7}$$

Text Line and Pixel Accuracies The PHR and TLL-DR metrics do not take into account false positives such as extra lines detected which do not correspond to any of the ground truth entities. [Fischer et al. 2014] presented an elegant way to deal with this problem; they introduced the Text Line Accuracy (Tla) and Text Line Pixel Accuracy (TL-PA) which treat the predicted/ground truth lines assignment problem by means of substitutions, deletions, and insertions. It performs the same one-to-one line assignment than [Y. Li et al. 2008] and after, the matching process, errors such as

insertion, splits, merges, and missed lines, are penalized. A line is correctly detected if it shares more than 90% of the pixels with the corresponding ground truth line. Hence, substitutions are matches that do not reach the given threshold. Deletions and insertions represent false negatives and false positives lines respectively. The matching value is calculated by the symmetric difference ($|A \triangle B|$) respect the intersection of two lines:

$$\frac{|R_i \triangle G_j|}{|R_i \cap G_i|} \geq 0.9 \ . \tag{D.8}$$

$$Acc_{T_\triangle} = \frac{N - S - D - I}{N} \ . \tag{D.9}$$

At pixel-level they defined the Text Line Pixel Accuracy (TL-PA):

$$Acc_P = \frac{N_P - S_P - D_P - I_P}{N_P} \tag{D.10}$$

Finally, as they claim, if you skip the insertion errors in this formula it leads to the correctness (Correctness $= \frac{N-S-D}{N}$) which corresponds with the TLL-DR and PHR.

Precision and Recall So far we have seen several evaluation techniques that are similar but with significant differences. TLL-DR and Tla based approaches require a one-to-one assignment, while MS based takes into account all the possible matches. As observed the pixel level evaluation always required from a one-to-one assignment to state the line that it belongs to. Put it in other words: a pixel cannot be claimed as correctly labeled if we do not know the line that must be assigned. One of the main drawbacks of the PHR is that it only takes into account the recall of the lines since it calculates how many pixels of the ground truth have been correctly labeled, but the recognition accuracy (or precision) is ignored. This measure could be adapted to extract the precision and hence the FM. [Fischer et al. 2014] solve these problems by adding insertions to the accuracy equation.

Our proposal generalizes the previous measures to compute the pixel precision and recall. Then we can translate the same idea at line level, which is very similar to the DR and RA. According to the best assignment S_o, it is possible to compute the *pixel recall* of paired lines as the ratio of the detected ground truth pixels by the

predicted ones. Analogously *pixel precision* is calculated by computing the ratio of predicted lines that correspond to the ground truth.

Equations D.11 and D.12 formalize the precision and recall between a pair of lines following the previous nomenclatures.

$$recall_{i,j} = \frac{P_{i,j}}{\sum_{j=1}^{N} P_{i,j}} = \frac{T(G_j \cap R_i \cap I)}{T(G_j \cap I)} . \tag{D.11}$$

$$precision_{i,j} = \frac{P_{i,j}}{\sum_{i=1}^{N} P_{i,j}} = \frac{T(G_j \cap R_i \cap I)}{T(R_i \cap I)} . \tag{D.12}$$

Finally, we aggregate the shared pixels $P_{i,j}$ of the whole assignment S_o, which, indeed, is the $Goodness$ previously introduced. Normalizing the $Goodness$ by the number of ground truth and predicted pixels give us the pixel and recall and precision respectively:

$$\begin{aligned} recall &= \frac{\sum_{k=0}^{max(M,N)} P_{k,S_o(k)}}{\#\text{gt}} = \\ &= \frac{Goodness(S_o)}{\#\text{gt}} = \\ &\sum_{i,S(i)}^{max(M,N)} \frac{T(G_{S_o(i)} \cap R_i \cap I)}{T(G_{S_o(i)} \cap I)} = H . \end{aligned} \tag{D.13}$$

$$\begin{aligned} precision &= \frac{\sum_{k=0}^{max(M,N)} P_{k,S_o(k)}}{\#\text{gt}} = \\ &= \frac{Goodness(S_o)}{\#\text{gt}} = \\ &= \sum_{i,S(i)}^{max(M,N)} \frac{T(G_{S_o(i)} \cap R_i \cap I)}{T(R_i \cap I)} . \end{aligned} \tag{D.14}$$

Hereby a balanced evaluation metric, the FM could be finally computed as:

$$FM_\beta = \frac{(1+\beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} . \tag{D.15}$$

$$FM = FM_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} . \tag{D.16}$$

Line level accuracy The precision and recall at line level require again from some thresholding criteria that states if a line is well detected or not. In this manner, we take into account the hits (correctly identified lines, eq: D.17).

$$\frac{P_i, S_o(i)}{\sum_{j=1}^{N} P_{i,j}} \geq 0.9, \quad \frac{P_k, S_o(i)}{\sum_{j=1}^{N} P_{i,j}} \geq 0.9 \ . \tag{D.17}$$

Dividing the $hits$ by the number of ground truth lines gives us the *line recall*, and if we do so by the number of predicted lines, we finally obtain the *precision*. Note that, the recall at line level corresponds on the TLL-DR previously introduced. Finally, the FM is computed as the harmonic mean between the precision and recall.

$$\begin{aligned} \text{precision}^{\text{line}} &= \frac{hits}{N} \ . \\ \text{recall}^{\text{line}} &= \frac{hits}{M} = \text{tll-dr} \ . \\ FM^{\text{line}} &= \frac{2 \cdot \text{precision}_{\text{line}} \cdot \text{recall}_{\text{line}}}{\text{precision}_{\text{line}} + \text{recall}_{\text{line}}} \ . \end{aligned} \tag{D.18}$$

To finish, it is worth to mention that these measures are just a generalization of the metrics proposed in [Y. Li et al. 2008]: the idea of the best assignment is maintained, but the precision of the prediction comes now into the game.

Opening Points Even though we have not used this measure during our experimentation, it is worth to mention it since it had been employed in some text line segmentation contests.

An *Origin Point* is defined as the location of the starting character of the first word of a line (Figure D.1). Indeed, this point is set in the left extreme of the baseline from the text line. This definition is very vague since we only indicate where a line starts. But, on some occasions, the primary challenge is to detect the number of lines and their begginings due to the possibility of merged and skipped lines. One big perk, though, is the easiness of the evaluation since we only need to compare a set of points. This metric has been used for some Text Line Segmentation contests; that is the case of the *Competition on Text Line Detection in Historical Documents* at ICDAR 2015.

The ground truth and the prediction are defined as set of *Opening Points*: Ground Truth Opening Points ($Q = Q_1, Q_2, Q_3, \cdots, Q_M$)

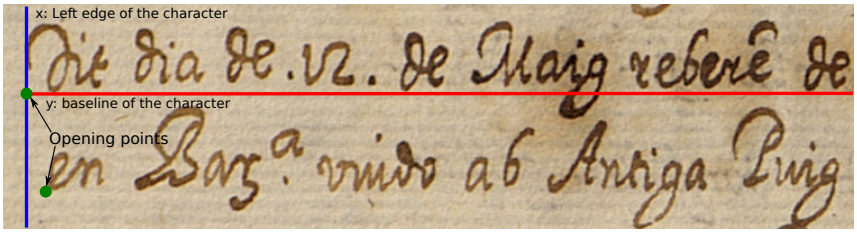


Figure D.1: Sample of the Opening Points of two lines. The opening point of a line is the intersection between the baseline and the left edge of the first character.

and the estimated Opening Points ($P = P_1, P_2, P_3, \cdots, P_N$). For getting the final scores, the two sequence are aligned. The alignment is performed by DP, minimizing the total Euclidean distances between matches. A hit is counted if two points (prediction-ground truth) are closer than a tolerance region R.

D.1.1 TLE metrics discussion

One can notice that MS and TLL-DR/PHR metrics are mostly equivalent. And in some cases, they work identically. Indeed, it is hard to find a practical case where the results differ. The main difference relies on the ground truth to predicted lines assignment. The TLL-DR based metrics compute the best assignment over all the possibles assignments, while in the MS the hits between lines are taken into account all the possible line pairs. Therefore, the later proposal is more reliable since it performs the one-to-one assignment which is more realistic and gives the possibility to compute the pixel level accuracy. As stated, the pixel level evaluation provide a detailed comparison between competitive methods since we could evaluate small differences regarding the frontiers of the prediction.

Another fundamental difference between both approaches relies on the way they compute a match between ground truth and the predicted match. The MS function counts an $o2o$ match if the intersection of foreground pixels of the lines are higher than 0.9 of their union. The PHR instead, checks if the precision and the recall separately are greater than 0.9 for the assigned match. In practice these measures are equivalent. Indeed, if a match follows the later criteria, it would follow the former one. Although one can come up with a very unlikely scenario where two matched lines have a pre-

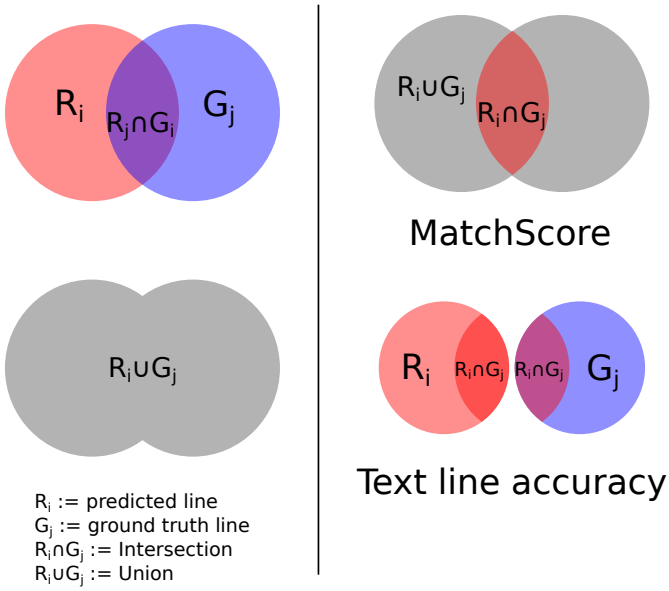


Figure D.2: Computation of hits for Match Score and text-line-level detection rate. The Match Score checks that the intersection $R_i \cap G_j$ is higher than the threshold. While Text Line Accuracy check that the intersection by the number of pixels in the ground truth and the prediction independently are over the threshold.

cision slightly lower than the threshold, and there will be counted as a match in MS metrics but not in TLL-DR. Figure D.2 illustrates the matching comparison using Venn's diagrams.

The precision and recall proposed metrics seem to us the best way of evaluating TLE because it gives us an idea about the errors committed. For example, with PHR and TLL-DR only the recall is taking into account, so no matter how many lines the method has guessed as soon as it detects the ground truth lines correctly. While adding the precision to the equation, shows the dispersion of the evaluated method, the same way DR and DA do in MS measure.

Computation of metrics Since all the possible pairs of predicted/ground truth, line scores must be computed the number of comparisons is $N \times M$. While the optimal assignment with the Hungarian Algorithm[2] presents an additional cubic cost. It is not a big problem since we do not deal with more than 100 lines in one document (and 100^2 and 100^3 order of operations are assumable by any modern computer). The main problem comes when evaluating the matches between two lines. Either in MS and Tla/PHR we need to check the number of foreground pixels overlapped between the predicted and the ground truth lines. Since the lines are usually defined as polygons shapes, the problem, now, relies on finding the foreground pixels inside of these regions. The evaluation procedure is evident in the bibliography, but the way it is computed in practice could be tricky.

Checking if a pixel is inside a polygon is not straightforward, even if we have a fast way of computing this value, we have a bunch of queries (one per each pair evaluated). For example, most of the methods, are linear with the number of nodes or edges and we have libraries that provide a fast and efficient implementation, but the line definition polygons have lots of edges, and doing the query for each of the foreground pixels makes the procedure heavier. We reduced the assignment problem to a polygon filling. We use a typical polygon filling operation (like the one in Paint) for creating a polygon mask: one corresponding to the ground truth area and the other for the predicted. Then we apply each mask to get the foreground pixels. If the two masks share the same bounding box, now it is straightforward to compute the intersection $G_j \cup R_i \cap I$ and the union $G_j \cap R_i \cap I$, and then we only need to count the black pixels to get the scores used for the evaluation. Figure D.3 illustrates this procedure.

D.2 Ground truth formats

Different formats for text line definition are discussed in this section. We have collected the ones we have used, and some we thought were worth to mention. Indeed, we are focusing on the text line ground truth and format definitions.

- **General purposes vs. Specific domain** Most of the document ground truth definitions include the line segmentation as

[2] https://en.wikipedia.org/wiki/Hungarian_algorithm

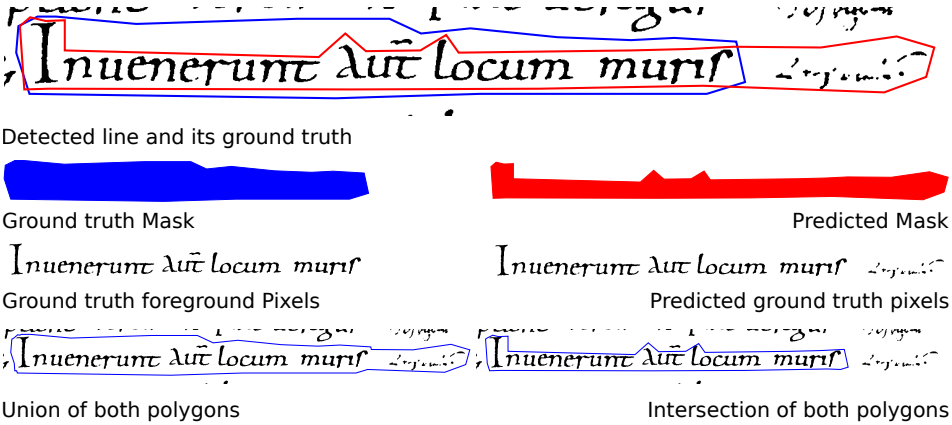


Figure D.3: Ground truth matching between one predicted line and its assigned ground truth. The blue polygon shows the ground truth area and the red one our prediction. Then two masks are extracted according to this areas and applied for the foreground pixels. Finally, we see the intersection areas and the union areas that are used for the evaluation metrics.

one of the many items coded. These elements correspond to other recognition stages like the layout definition or the final transcription.

- **XML/JSON and other schemas** Some of the formats included a well-defined ontology that allows interchange and data checking. Other formats just rely on the line definition using pixel encoding or polygon coordinates in raw files.
- **Evaluation versus transcription formats** Some formats are designed to be used by experts or OCR engines(*TEI, IIF*) and others are more focused on the evaluation process: *PAGE, Pixel Encoded*.

PAGE The *Pattern Recognition and Analysis* (PRimA) research group at the University of Salford has developed and extended the PAGE XML format schema to add and relate the many document description elements. The creators of PAGE [Pletschacher et al. 2010] claimed that there were several representation formats, but each of them is designed to one of the individual transcription stages. Therefore it is intended as a general purpose format that allows several documents and a vast range of properties. Besides, PAGE is more than a XML schema, it provides a set of tools to deal

with other related operations: converters, viewers, validators, even C++ and java libraries[3].

The format is widely used by some groups and contests, for example, the HisDoc project[4] provide access to a ground truthing tool that generates document definitions following this format. The *IAM Historical Database* which we have worked with is also available in the PAGE format.

Advantages:

- Suitable framework and evaluation tools provided for several platforms and programming languages.
- Clear data structure with a XML validated schema.
- It is capable of providing information not only on text line segmentation and Layout Analysis but transcriptions and more document preprocessing stages.

Handicaps:

- It is a very general purpose definition. Even the schema is complete and well define, one needs to perform an specific parsing of the document to extract the line definitions.
- The XML data overload with redundant tags.

```
<PcGts xsi:schemaLocation="http://schema.primaresearch.org/PAGE/gts/
    pagecontent/2013-07-15 http://schema.primaresearch.org/PAGE/gts/
    pagecontent/2013-07-15/pagecontent.xsd" pcGtsId="">
  <Metadata>
    <Creator>hao.wei@unifr.ch</Creator>
      <Created>2014-mars-12 15-09-08+0000</Created>
        <LastChange>2014-Jul-31 14-28-32+0000</LastChange>
      <Comments/>
  </Metadata>
  <Page imageWidth="1664" imageHeight="2496" imageFilename="csg562
      -003.png">
    <TextRegion type="text" id="1396720108" custom="0" comments="">
      <Coords>
        <Point x="213" y="963"/>
        <Point x="218" y="983"/>
        <Point x="224" y="1011"/>
          ...
        <Point x="222" y="960"/>
      </Coords>
    </TextRegion>
```

[3] http://www.primaresearch.org/tools/PAGELibraries
[4] http://diuf.unifr.ch/main/hisdoc/divadia

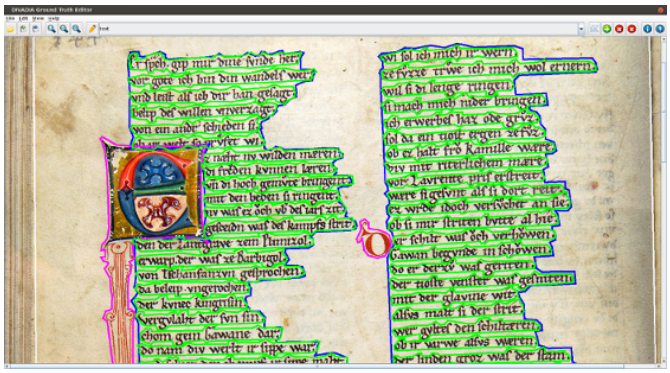


Figure D.4: Screenshot of the *Divadia Ground Truth Editor* and the visual text layout and text line segmentation of one page from the Saint Gall database.

```
    <TextRegion type="textline" id="3681944" custom="0" comments=""></
        TextRegion>
    <TextRegion type="textline" id="5037281" custom="0" comments=""></
        TextRegion>
    <TextRegion type="textline" id="8361707" custom="0" comments=""></
        TextRegion>
    <TextRegion type="page" id="3999666" custom="0" comments=""></
        TextRegion>
    </TextRegion>
  </Page>
</PcGts>
```

Pixel encoded TLE is usually evaluated taking only into account foreground pixels. That makes sense since it is hard to describe the exact boundaries of the lines in the space areas between them.

The image could be coded as a 2D matrix, where the label 0 indicates background and the integer values in the range $[1, N]$ describe the pixel assigned line (N is the total number of lines). If the number of lines expected is less than 255 it can be easily saved as 8 bits grayscale image. If it is not the case, we can use other matrix portable formats like *numpy*, *matlab*, or our internal format in *APRIL-ANN*. This line definition was used at the *ICDAR 2009 Handwritten Segmentation Contest, ICFHR 2011 Handwritten Segmentation Contest and the ICDAR 2013 Handwritten Segmentation Contest* [B. Gatos et al. 2009; B. Gatos, Stamatopoulos, et al. 2010; Stamatopoulos et al. 2013].

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0
1	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1	0	0	1	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	1	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1	0	0	0	1	0	0	1	0	0	1	0	0	1	0	0	0	1	0	0	0
1	0	1	0	0	0	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0	0	0	0	1	0	0	0
0	1	0	0	0	1	0	1	0	0	1	0	0	0	1	0	0	0	1	1	0	0	0	0	1	0	0	0
1	0	1	1	1	0	0	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	2	0
0	2	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	2
0	2	0	0	2	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
0	2	0	0	2	0	0	0	0	0	0	2	0	2	2	2	0	0	0	2	2	0	0	0	0	0	0	2
0	2	0	2	0	0	0	0	2	0	0	0	2	0	0	2	0	0	2	0	0	2	0	0	0	2	2	0
0	2	0	2	0	0	0	2	2	0	0	0	2	0	0	2	0	0	2	0	2	0	0	0	0	2	0	0
0	0	2	0	0	0	2	0	2	0	0	2	0	0	0	2	0	0	0	2	2	0	0	0	2	2	0	0
0	2	0	2	2	2	0	0	2	2	2	0	0	0	0	2	2	2	2	0	0	2	2	0	2	2	2	2

Table D.2: Illustration of the pixel level line encoding.

Advantages:

- Could be encoded as a grayscale image, and it could be visually checked with a conventional image viewer.
- Only foreground pixels are tagged. That avoid problems related with lines boundaries definition.
- Fast computation for most of the evaluation metrics.
- Almost all formats can be converted to this codification (albeit they would lose information). This is the lowest level used by TLE evaluators.

Handicaps:

- The format does not allow annotation or transcriptions. That should be added to new collections.
- It has some problems to be supervised or modified because there are not line frontiers or anchors points that ease the supervision. Some approaches like Voronoi could extract the boundary lines from the pixel definition, but it does not guarantee to be easy to supervise or modify.

Sparse pixel codification Since the codification is done only on foreground pixels, it is possible to use a sparse representation based on indexed pixels. In this case, for each text line, we define the list of foreground pixels by coordinates that belong to that class.

Each line of the text file includes the foreground pixels of that text line. This definition allows getting all the pixels of a line easily, other formats instead, require running over the whole image to assure that we got all the pixels of one line. Nevertheless, the 2D representation image is a more explicit codification than the indexed.

Medieval Manuscript Layout Model [Baechler and Ingold 2010] introduced a Layout Model for Historical Manuscripts. A text line definition associated was proposed. It is worth to mention this work since it addresses the problem of ground truthing text lines on historical documents.

Here it is proposed a Layout Model specialized on Medieval Documents. And it separates the parts of the scanned pages on 4 layers: degradation, text, comments, and decoration. Moreover, the line boundaries are defined using isothetic polygons.// Isothetic polygons are an elegant solution for the line segmentation ground truth: the lines are formed only by vertical and horizontal segments[5]. In the proposed XML schema, the polygon is described by the upper and lower isothetic lines. The extremes of both lines are assumed to be joined for vertical straight lines closing the polygon area. The text line model is coded on the text layer. Also, it is possible to compute tokens for each line like words, characters, and glyphs.

Advantages:

- This is an specific model for Medieval documents. It provides the minimum requirements to cover the properties of this kind of documents. Hence it avoids unnecessary generalization and complexity of the DOM XML Schema. From a developing point of view, it simplifies the understanding and posteriors parsing operations.
- It also provides an hierarchical structure from layers to nested elements within each of the layers.

[5] https://en.wikipedia.org/wiki/Isothetic_polygon

- Isothetic polygons use straight vertical/horizontal lines. The frontiers of the polygon are clear, easy to compute and avoid aliasing problems or numerical errors while calculating the interpolated polygon lines.

Handicaps:

- The layers proposed deal with most of the cases from medieval documents. But one can always find unexpected information; it is missing some more generic class with a personalized type.
- In this format, they decided to use the isothetic polygon definition, besides different types of polygons can be easily included. The upper and lower line definition of the polygon makes sense for text lines, but in decoration or figures, it is not clear what is the upper and lower contour.
- The isothethic definition of the line is not natural for the human eye, and also it is a bit uncomfortable to supervise.

SVG polygons We have seen XML schemas for encoding the document associated information. But, we already have standardized, well-known and widely used XML definitions that allow drawing elements (lines, polygons, shapes, embedding images). That is the case of the widely XML-based image format *Scalar Vector Graphics (SVG)* [6].

For the text line definition, we can use the `polyline` tag that allows creating polygons, usually through a set of coordinates. The coordinates could be absolute or relative, which each node has the translated position respect to the previous one.

Advantages:

- It is a XML based format that is widely used and includes lots of useful tools.
- It allows to embed or link the underlying document image. Thus it is possible to visualize the text line definition with an appropriate image viewer (most of the web modern browsers support SVG).
- Easy to supervise, since the edges of the lines, are preserved with the appropriated tool it is possible to modify or add new edges to the polygons.

[6] More information about SVG in `https://en.wikipedia.org/wiki/Scalable_Vector_Graphics` and `http://www.w3schools.com/svg/`.

Handicaps:

- There are several ways of defining the `polyline` coordinates, not only straight segments, it can include shapes and curves. This makes harder to check the pixels inside of the polygon. Usually, we have to restrict the polygons segments to straight lines to ease the next evaluation.
- The SVG format was not mainly conceived for text line codification. Instead, it is a general purpose drawing library. What we are doing here is taking advantage of some of its capabilities for defining our properties. But mostly we are drawing lines and polygons on a canvas.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
 "http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg xmlns:svg="http://www.w3.org/2000/svg"
xmlns="http://www.w3.org/2000/svg" version="1.0" width='2690' height
    ='3794
' id='svg_document' style="display:inline">
  <g id="Volum_069_Registres_0004.png">
    <image xmlns:xlink="http://www.w3.org/1999/xlink" id="BgImageID" x
        ="0" y="0" width="2690" height="3794" xlink:href='
        Volum_069_Registres_0004.png'/>
    <g id="lines">
      <polyline class="line_0" points="198,544 257,544 ... 2121,638
          2121,639" style="fill:none;stroke:green;stroke-width:5"></
          polyline>
      <polyline class="line_1" points="300,825 409,805 ... 2151,770
          2160,763 " style="fill:none;stroke:green;stroke-width:5"></
          polyline>
      ...
      <polyline class="line_27" points="373,3600 481,3600 ... " style=
          "fill:none;stroke:green;stroke-width:5"></polyline>
    </g>
  </g>
</svg>
```

Polygon coordinates One of the easiest ways of line encoding is just to annotate the coordinates of each the polygon surrounding a line. This can be as easy to note in a text file the coordinates of the surrounding area of the text line. So in this case, we will have a polygon per line with each coordinate, where the last point in the sequence it is joined to the first one closing the area.

Advantages:

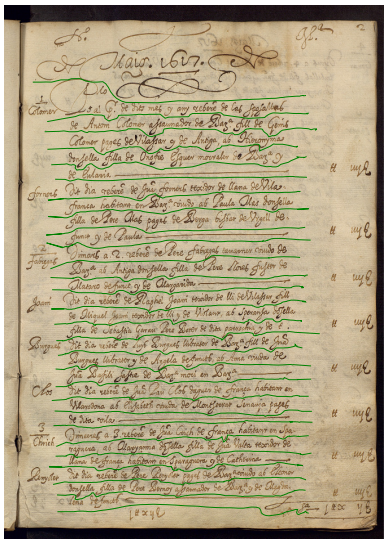


Figure D.5: SVG visualization for the text line ground truth for a page from the ESPOSALLES Marriage Records database.

- Very simple definition but powerful enough for describing lines.
- It is easy to parse and speeds up the collaboration between researchers since the definition is straightforward: polygon coordinates. Not ambiguities, overcharge data like *XML* and other structured formats.
- Could be converted in other formats: PAGE, SVG, labeled pixel, or even *TEI*.
- Easy to supervise, since the edges of the lines, are preserved, and with the appropriated tools it is possible to modify or added new edges to the polygon definition.

Handicaps:

- Extremely simple definitions does not allow much more information than the line boundaries. No layout definition neither transcription.
- It does not follow any scheme, which difficult to check the correctness of the description.

Text Encode Initiative

As reported in their main site,

> The Text Encoding Initiative (TEI) is a consortium which collectively develops and maintains a standard for the representation of texts in digital form. Its chief deliverable is a set of Guidelines which specify encoding methods for machine-readable texts, chiefly in the humanities, social sciences and linguistics. Since 1994, the TEI Guidelines have been widely used by libraries, museums, publishers, and individual scholars to present texts for online research, teaching, and preservation. In addition to the Guidelines themselves, the Consortium provides a variety of resources and training events for learning TEI, information on projects using the TEI, a bibliography of TEI-related publications, and software developed for or adapted to the TEI.

Text Encode Initiative (TEI) provide a broad set of possibilities to encode several media. Hence it is possible and widely used the TEI definitions for historical manuscripts. The guidelines include the option to define lines and their transcription.

```
<TEI>
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>Title</title>
      </titleStmt>
      <publicationStmt>Publication</publicationStmt>
      <sourceDesc>Source Description</sourceDesc>
    </fileDesc>
  </teiHeader>
  <facsimile>
    <surface xml:id="surface1" ulx="0" uly="0" lrx="1258" lry="1903">
      <graphic url="http://digi.ub.uni-heidelberg.de/diglitData/image/
          cpg148/4/006v.jpg"/>
    </surface>
    <surface xml:id="surface1" ulx="0" uly="0" lrx="1258" lry="1903">
      <graphic url="http://digi.ub.uni-heidelberg.de...007v.jpg"/>
      <zone xml:id="Area0" ulx="45" uly="134" lrx="330" lry="1138">
        <line xml:id="Area0l0" ulx="115" uly="149" lrx="269" lry="189"
            />
        <line xml:id="Area0l1" ulx="115" uly="190" lrx="303" lry="217"
            />
        <line xml:id="Area0l2" ulx="114" uly="222" lrx="271" lry="247"
            />
```

```
      </zone>
      <zone xml:id="Area1" ulx="57" uly="1131" lrx="713" lry="1737"/>
    </surface>
  </facsimile>
  <text><body>
    <div facs="surface1">
      <p><s>
        <title>New Page</title>
      </s></p>
      <div>
        <pb facs="http://digi.ub.uni-heidelberg.de...006v.jpg"/>
      </div>
    </div>
    <div facs="surface1">
      <p><s>
      <title>New Page</title>
      </s></p>
      <div>
        <pb facs="http://digi.ub.uni-heidelberg.de/diglitData/image/
            cpg148/4/007v.jpg"/>
        <p facs="#Area0">
          <l xml:id="l0" facs="#Area0l0"/>
          <l xml:id="l1" facs="#Area0l1"/>
          <l xml:id="l2" facs="#Area0l2"/>
        </p>
        <p facs="#Area1"/>
      </div>
    </div>
  </body></text>
</TEI>
```

Advantages:

- It allows line definitions and more complex characteristics of the documents. zoning, glyphs and more semantic information.
- Very complete schema with support to lots of document variants and issues.
- Templates and starter examples are provided.

Handicaps:

- Even though the clear guidelines and the completeness of the formats, it seems (as seen in the example) a bit tedious to dive in the XML schema for doing image transcription and zone segmentation.
- The complexity of the schema could be overkilling since it is a very general codification.

International Image Interoperability Framework (IIF) As its name suggests, it aims to provide interoperability between different image repositories. Thus, a set of tools, standard definitions, and operations over images and image-based data are supplied. Citing the API abstract:

> The IIIF Image API specifies a web service that returns an image in response to a standard HTTP or HTTPS request. The URI can specify the region, size, rotation, quality characteristics and format of the requested image. A URI can also be constructed to request basic technical information about the image to support client applications. This API was conceived of to facilitate systematic reuse of image resources in digital image repositories maintained by cultural heritage organizations. It could be adopted by any image repository or service and can be used to retrieve static images in response to a properly constructed URI.

W.r.t the Text Line Encoding, IIIF applies the concept of segments that define parts on the document[7]. It allows defining rectangular bounding boxes or other nonrectangular segments using SVG definitions.

The data interchange is performed using the Javascript Object Notation (JavaScript Object Notation (JSON)). Following we show a sample of a transcribed document, the code has been taking analyzing the XHR web service request from one of the *Biblissima* [8].

Lately it has been converted to IIIF applying an XSL transformation from TEI to JSON-LD[9].

```
"@context": "http://www.shared-canvas.org/ns/context.json",
"@id": "http://demos.biblissima-condorcet.fr/iiif/metadata/
    BL_Add_10289/list/transscript_8v.json",
"@type": "sc:AnnotationList",
"resources":[

    {
    "@id":"http://demos.biblissima-condorcet.fr/iiif/metadata/
        BL_Add_10289/Annotation/8v_line449.json",
    "@type":"oa:Annotation",
    "motivation":"sc:painting",
```

[7] http://iiif.io/api/presentation/2.0/#advanced-association-features
[8] http://demos.biblissima-condorcet.fr/roman-bl-caen/m1/
[9] https://github.com/stefaniegehrke/TEI-2-SC

```
    "resource":{
    "@type":"cnt:ContentAsText",
    "chars":"De la forest a feit areine",
    "format":"text/plain",
    "language":"fr-FR"
    },
    "on":"http://sanddragon.bl.uk/IIIFMetadataService/canvas/folio-8v.
        json#xywh=1700,500,3800,208"
    }
 ,
    {
    "@id":"http://demos.biblissima-condorcet.fr/iiif/metadata/
        BL_Add_10289/Annotation/8v_line450.json",
    "@type":"oa:Annotation",
    "motivation":"sc:painting",
    "resource":{
    "@type":"cnt:ContentAsText",
    "chars":"Entor le mont, et bele et pleine.",
    "format":"text/plain",
    "language":"fr-FR"
    },
    "on":"http://sanddragon.bl.uk/IIIFMetadataService/canvas/folio-8v.
        json#xywh=1700,708,3800,208"
    }
 ,
}
```


Diva Services The Diva Services [Marcel et al. 2016] is an initiative to provide a set of RESTful webservices using JSON schemas, in a similar way than IIIF does. The main difference is that Diva Services provide algorithms and techniques that can be used by other external tools: line extraction, layout analysis, DIA, and OCR (interest points, polygonization. . .).

The primary objective is to normalize the interoperability between tools and algorithms. The clearest example is the text line extraction methods. In this scenario, one tool could use a text line method developed by somebody else using a web server. Moreover, externalization of some stages allows powerful servers to assume the computation of heavier algorithms. And it avoids to re-implement or compiles the algorithms for different platforms and devices.

In the case of text line extraction methods, the RESTful web service response uses a JSON schema that defines the lines, which can be seen as a ground truth definition of the text lines.

Next, it is shown the XHR response for a text line histogram based algorithm; we can see that the lines are contained in rectangles.

Nevertheless, the response could use polygons for the text line segmentation.

```
{"output":{},"highlighters":
  [{"rectangle":
     {"segments":[[312,653],[1970,653],[1970,700],[312,700]]}},
   {"rectangle":
     {"segments":[[669,731],[1663,731],[1663,749],[669,749]]}},
   {"rectangle":
     {"segments":[[297,764],[1618,764],[1618,801],[297,801]]}},
   {"rectangle":
     {"segments":[[490,836],[1918,836],[1918,887],[490,887]]}},
   {"rectangle":
     {"segments":[[464,927],[1914,927],[1914,975],[464,975]]}},
   {"rectangle":
     {"segments":[[320,1021],[1985,1021],[1985,1070],[320,1070]]}}
  ]
}
```

Following the corresponding POST request from the server:

```
{
  algorithm: {name: "Histogram Based Text Line Segmentation"},
  highlighter: {type: "rectangle", closed: true,
   segments: [[250, 611], [250, 2830], [2047, 2830], [2047, 611]]},
  image: {_id: "5625010c966f808d22e463af",
          clientName: "csg562-063.png",
          extension: "png"},
  inputs: {}
  start: "2015-10-19T14:43:43.419Z"
  started: true
}
```

Advantages:

- Nowadays the interest of using web services (specially RESTful) is growing in all the researching: economics, biology, forecasts, social networks. Use them as standard interchange format is a significant advance for the libraries and transcribers.
- In the case of Diva Services, the initiative allows to various teams and research groups to provide their algorithms as web services which allow to apply them to new tasks easily and compare the results.

Handicaps:

- These formats are thought for interchange and not for storage. What one could see here is the redefinition of XML and other schemas into JSON.

- It is needed to standardize the algorithms schemas on the DIVA Services since some of the algorithms are parameterized.

Others We apologize for not including other formats, and we are sure we have forgotten to mention a bunch of several text definitions.

Some of the formats that could not been reviewed in this format exploration have been the *ALTO* (Analyzed Layout and Text Object) format[10], *hOCR*[11] or *Pink Panter* [Yanikoglu et al. 1998].

Humanist and historical communities From an humanistic point of view, just say a few words about the very necessity of providing reliable formats and tools to standardize the transcription procedure. During this chapter, we focus on TLE, but when talking about ground truth and coding, the corresponding data formats must also be reliable for the following human transcription, correction, and annotation. If the whole transcription process is perfect no human supervision is required, but w.r.t to complex ancient/historical documents the recognition engines are still far from perfect. In fact, the final users of this tools are mainly humanists and experts which are aiming to transcribe as much as documents as possible. When one dives a bit in this community, it is surprising that some of the transcribers use a side-by-side document image and the text editor to perform the transcription. In other cases the codification of the transcription on the XML based TEI encoding which is popular among humanists: the expert opens an XML editor and works directly on the XML definition. If the alignment is noted, they even take coordinates from an image viewer/editor. This procedure leads to a lot of errors and unnecessary human overwork. For this matter well defined and logical models as well as robust are desirable for the humanist community.

[10] https://www.loc.gov/standards/alto/techcenter/layout.php
[11] https://en.wikipedia.org/wiki/HOCR

Appendix E

Text Reference Lines (soft) ground truth generation

One of the key points of this PhD Thesis is the concept of Interest Point (IP) which has been used in TLE and TLE. The same ideas have been followed in both tasks: first, we use the IPs for extracting the reference lines and, then, we determine the Main Body Area (MBA) of the text line. In the former scenario, the MBA is used to extract the orientation and areas of the lines. In the later, the MBA is used for a non-uniform text height normalization.

This IP extraction and classification requires from a supervised ground truth, which is obtained by supervising and classifying LEPs.

In this Appendix we show the supervision of LEPs/IPs and the related data generated to assist our models. We refer to it as "soft" ground truth since the IP classification task is not a proper application. It is an intermediate step that we employ to build models that used in further tasks.

E.1 Interest Point (IP) supervision

In Section 5.3 we showed how the IP are extracted and classified. In TLE the Local Extrema Points (LEPs) are extracted from a full page and in TLN the LEP are used at line level instead. There are two main differences to separate the two process:

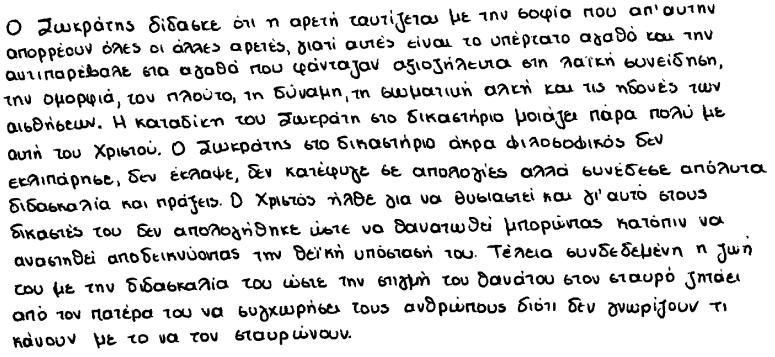


Figure E.1: Sample of ICDAR 2013 Handwritting Segmentation Contest.

- It is easier, as we will see, to supervise the points at line level than the in the whole page. So for the supervision stages, we will separate the entire process in the two stages.
- When classifying LEPs, the classes are different for the line and page scenarios. At the page level, we add new classes that we could in the line based classification since the input lines have been separated than decorations and noises.

Even though the primary objective is to extract the supervised points for the whole page, in TLE, the reference lines are defined at line level. Hence, it is more intuitive to supervise one line at a time. Therefore, we split first the pages into lines, then compute the IP and finally, this information is translated back to the full page.

Cut lines The first step is to extract the lines from the page. We use for this purpose the text line definition in the ground truth. Each line is cut and pasted in a new blank image of the size of its bounding box. Even if the bounding box contains traces or parts from other lines, we copy only the foreground pixels of the current line.

Initial IP Classification Once the lines have been cut, we need to supervise them. For that, it is required and initial classification of the points and then correct the mistakes. The classification of points at text line level was presented in [Gorbe-Moya et al. 2008]. There are two main contributions in this PhD Thesis with respect to the original IP framework:

1. Two new classes are added in our contribution: ignore and dirty.
2. The models have been improved by including CNNs.

The idea is to have an initial classification of the points. We have three options:

- Local Extrema or Naïve classification. No prior classification is performed. Only the extrema points are extracted, then the supervisor starts with a non-labeled set of points.
- Heuristic classification. In this case, we do not rely on ML methods. We had tried to apply histogram based rules. The results are not perfect, and it requires posterior supervision.

  The heuristic classification is performed by means of horizontal projection histogram. For that, the upper and lower baselines are estimated using a fixed height according to the frontiers of the histogram. Then the LEPs are classified according to the proximity to these lines.
- Transferred learning. If another model is trained previously, it can be used for the initial classification. For this purpose would be great to have a range of models for different styles and fonts and use the one who has higher similarity.

This classification step is crucial, and a good initial set can save human effort. Moreover, a weak classification could be counterproductive since the expert will have to remove lots of misclassified points.

Note that at text line level the *dirty* class barely appears during the supervision, but later we will add this IP as the points on the page that are not in a text region.

Text line Bootstrap Then a bootstrap supervision process is followed. The underlying idea is to supervise a few set of lines, then train a first model, generate a new set of IPs and supervise new lines taking into account the new supervision. And we can do this iteratively to speed up the process. With this into account, the first question that comes up is about how many lines supervise in each iteration. It relies on several factors: available for supervision, accuracy of the first segmentation, the type of data. For instance, we can set fixed amount of time for each supervision step. Let us say 1 hour approximately, in that time we classify as many points as possible and after we retrain the classifier with the new points. Now, we continue supervising for another hour, if the things go as expected the speed of supervising is supposed to improve because fewer corrections are committed. Then another retraining of the model and continue supervising more pages, since we reach the number of lines that can assure us a good performance in the task we are dealing with.

Translate points to page In this stage, the points supervised are brought back to the original full page.

The methods exposed in this work on the beginning are using a receptive field related to the pixel or interest point we want to classify. If we train our ANNs using only the lines, the receptive field will include only information of the current line and it will not have more surrounding information, which is critical when computing line frontiers.

The straightforward approach is to translate the supervised IPs directly to the page. Then we calculate the rest of the LEPs of the page and mark them as dirty. However, this approach has some issues that we had to deal with:

1. The IPs extracted at text line level are not detected on the main page or are detected in another position. This artifact happens often:
    - The line cut intersected some stroke, and fake LEPs are detected.
    - The LEP computation has a special treatment when we have a wide local maxima/minima stroke (illustrate in Figure E.5), where we set up a gap of n pixels between points

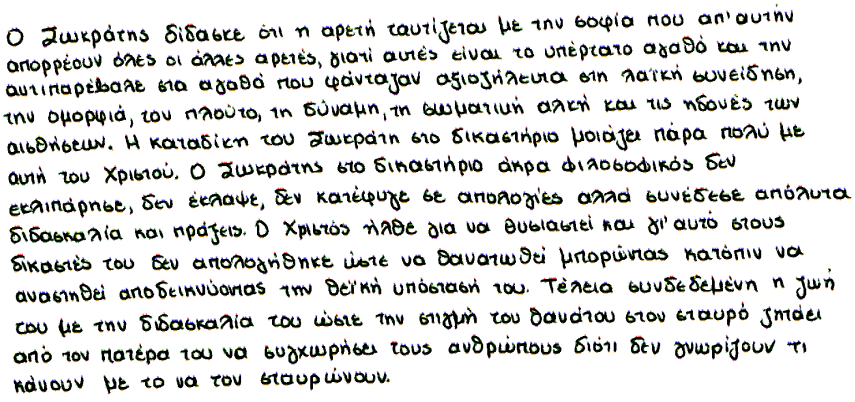


Figure E.2: Sample of the Interest Points translated into the full page.

to avoid a continuous line of pixels. These LEPs could have different positions on the line and the page.

2. There are LEPs that are inside of the text but it has been not detected on the text line splitting.

To overcome problem 1, we had to compute all the LEPs at the page level and then tag the new points if they match with one of the supervised (using a neighboring window of 1 or 2 pixels).

When dealing with problem 2 we can forward this error to the supervisor in the next stage or just use the layout ground truth to check the area which is contained (text/non-text).

Figures E.2, E.3 shows the results after moving the points of the ICDAR 2013 corpora to the full page. Note that, this dataset is clean and has only text. Figure E.4 shows the `noise/non text` points extracted over the whole page.

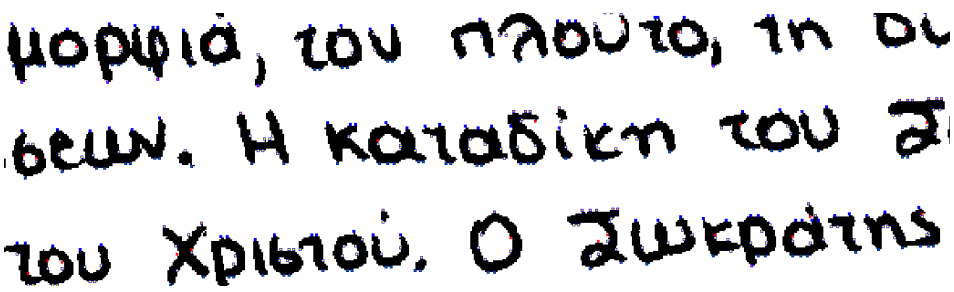


Figure E.3: Sample of the Interest Points translated to the Page Level (Detail).

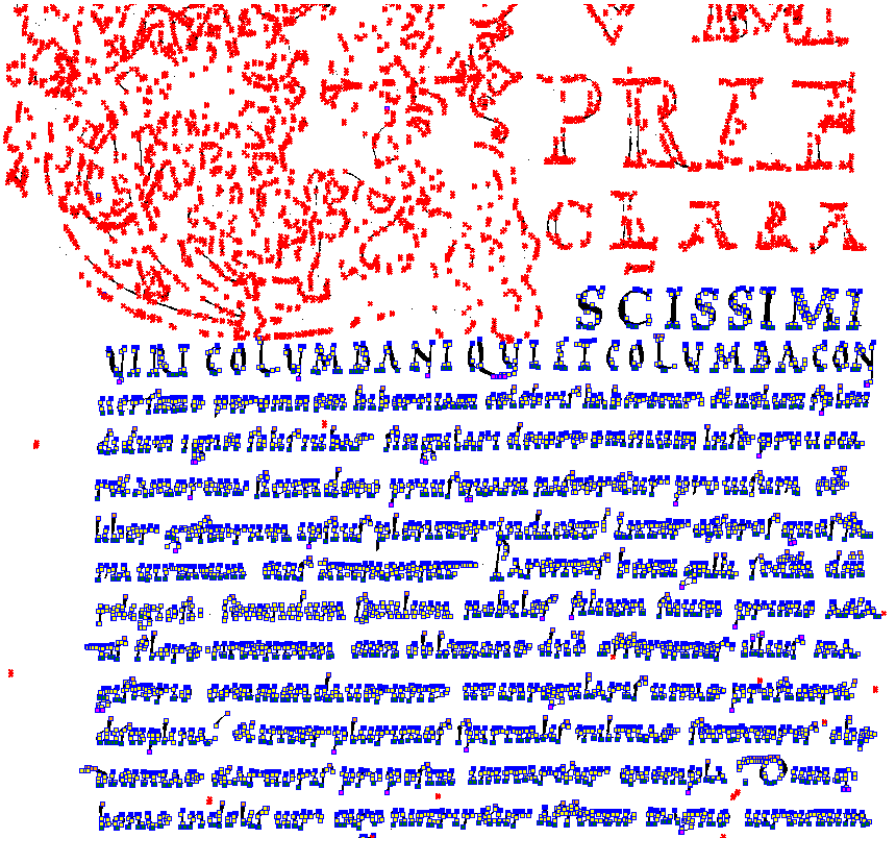


Figure E.4: Sample of the dirty points added.

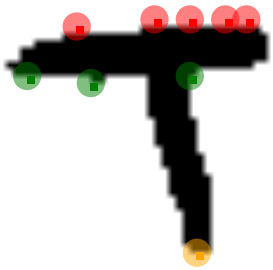


Figure E.5: Example of wide Local Maxima Strokes. Points are distributed along the streak line.

Full Page Supervision After translating the points to the page, we can observe in Figure E.4, we can see some of the problems described in the previous stage. So in this step, an expert could supervise some mistakes. The supervision effort in this stage is minimal.

E.2 Main Body Area supervision

In this PhD Thesis we explored to classify pixels into MBA without relying on IP.

For this purpose, it will be more suitable to have a ground truth directly of the MBA. Where for an image or line, we have two zones: MBA and not MBA. The codification, indeed is very straight forward, a binary image where 1 means that the pixel belongs to the MBA and 0 the rest.

The supervision at the pixel level is more complicated and even subjective than the IP. Nevertheless, the MBA is delimited by the reference lines (mean line and baseline) and we used IP for delimit the text reference lines. Thus, it is possible to generate the MBA from the IP. Since we have performed already the IP supervision, we use the same ground truth to create the MBA images.

The procedure is performed as follows and it is illustrated in Figure E.7:

1. LEPs are extracted for each of the lines.

2. The points are classified and then supervised as depicted in Section E.1.
3. The mean line and baseline points are joined to generate the reference lines.
4. The area between the mean line and baseline is marked as MBA.
5. The line MBA ground truth is translated to the page.

Ironically, for the IP classification we need the extract the IPs class from the reference lines by human supervision. Now we are reversing the procedure, we extract the reference lines from the points in order for marking the MBA.

E.3 Case of study. Bootstrap

In this section, we illustrate the supervision procedure and the effort invested while supervising the data from the ICDAR 2013 Handwritten Segmentation Problem [Stamatopoulos et al. 2013]:

1. The text lines are cut and extracted from the available ground truth[1].
2. The trained model from Saint Gall corpora is used for the initial classification. Unfortunately, the results are not good enough, and we need to invest some time to supervising the lines.
3. After less than 1 hour of supervising, we could supervise the 14 lines corresponding to the first page. That is 5417 points in total on an average of 2 minutes per line. Unfortunately, most of the points have been relabeled (around 40% of the points).
4. A new ANN model is trained, using the new data.
5. The rest of lines are classified with the new model.
6. Now we start to supervise again. As we can see in figure E.8, the model is still not good enough, and we have to continue supervising points.
7. In this new iteration, we supervise 2 pages:
   - A total of 31 lines.
   - Total set of 11619 points.

[1] http://users.iit.demokritos.gr/~nstam/ICDAR2013HandSegmCont/

- 3238 points are relabeled (28%) .
- Around 54 minutes, 104 seconds on average per line.

8. We retrain the new model, using this time 2 pages as train and one as tuning set. Then we regenerate the points with this new model.
9. It is time to supervise more pages, in this case, 4. As we can see the corrections and the supervising time are decreasing in each iteration.
   - A total of 90 lines.
   - Total set of 24048 points.
   - 7132 points are relabelled (29%) .
   - Around 135 minutes, 90 seconds on average per line.

With this data, we can train our models for text line extraction. In around 3 to 4 hours of supervising effort we got:

- 7 pages.
- 120 lines.
- 42738 labeled interest points.

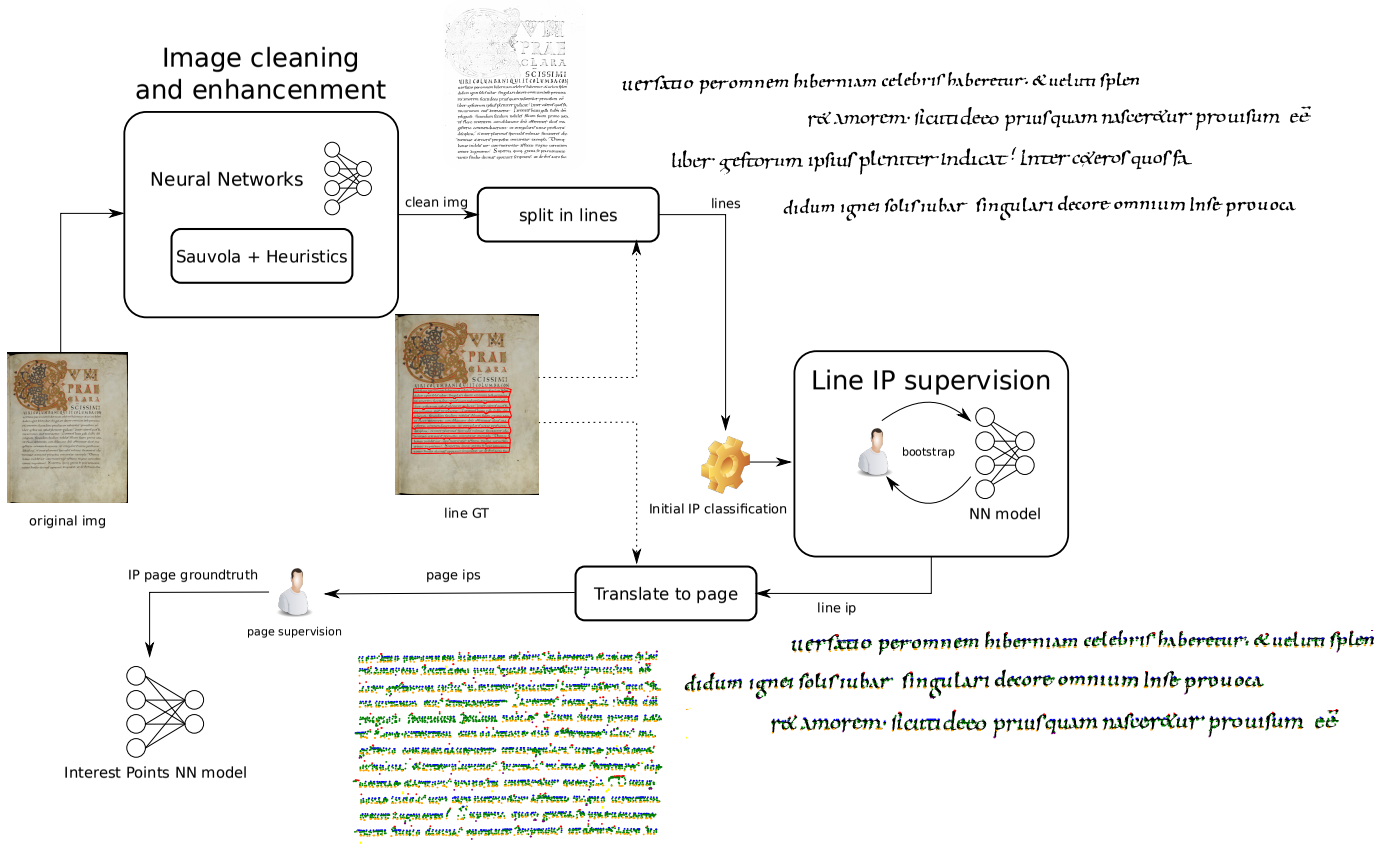


Figure E.6: Interest Point ground truth generation. The schema shows the supervision workflow. 1) The image is cleaned. 2) The IP are supervised at line level, 3) Translated to the page. 4) Final page supervision.

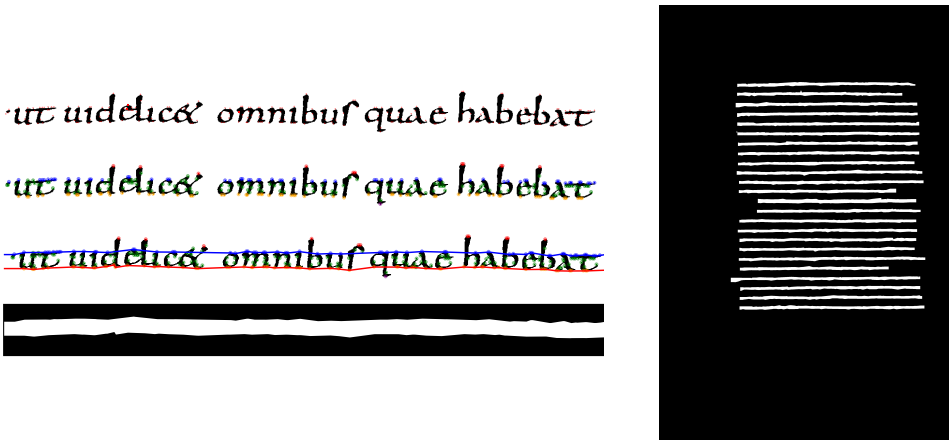
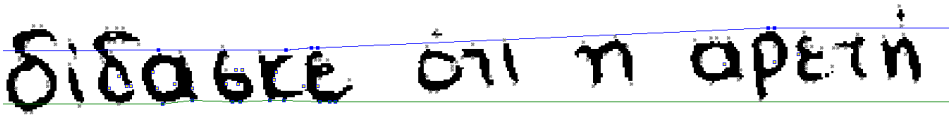
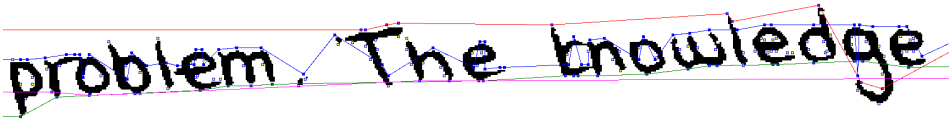


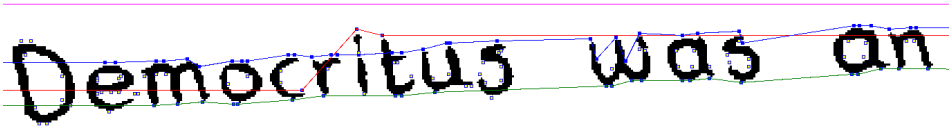
Figure E.7: Main Body Area computation from Interest Points. In the left part, we have the procedure for the mark the MBA at line level: a) Local Extrema Points, b) Interest Points classified, c) Meanline and baseline, d) MBA of the line. In the right part, the MBA of all the lines is translated to the full page.



(i) Initial classification



(ii) Sample with the model after supervising one page.



(iii) Sample with the second model. Using one page for training and another for validation.

(iv) Sample with improved model: 3 pages for training, 2 for validation.

Figure E.8: Bootstrap process for text line IPs classification.

References

Abdel-Hamid, O., A. R. Mohamed, H. Jiang, and G. Penn (2012). “Applying convolutional neural networks concepts to hybrid NN-HMM model for speech recognition”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 4277–4280 (cit. on p. 39).

Adelantado-Torres, J. L., J. Pastor-Pellicer, and M. J. Castro-Bleda (2014). “Una aplicación móvil para la captura y mejora de imágenes de textos”. In: *V Jornadas TIMM* (cit. on p. 19).

Afzal, M. Z., J. Pastor-pellicer, F. Shafait, T. M. Breuel, A. Dengel, and M. Liwicki (2015). “Document Image Binarization using LSTM : A Sequence Learning Approach”. In: *Third International Workshop on Historical Document Imaging and Processing*, pp. 79–84 (cit. on pp. 18, 97, 98).

Amodei, D., R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu (2015). “Deep Speech 2: End-to-End Speech Recognition in English and Mandarin”. In: *Jmlr W&Cp* 48, p. 28 (cit. on p. 38).

Amudha, J., N. Pradeepa, and R. Sudhakar (2012). “A survey on digital image restoration”. In: *Procedia Engineering*. Vol. 38, pp. 2378–2382 (cit. on p. 16).

Antonacopoulos, A. and A. C. Downton (2007). “Special issue on the analysis of historical documents”. In: *International Journal on Document Analysis and Recognition* 9.2, pp. 75–77 (cit. on p. 16).

Arruda, A. W. A. and C. A. B. Mello (2014). “Binarization of Degraded Document Images Based on Combination of Contrast Images”. In: *14th International Conference on Frontiers in Handwriting Recognition*, pp. 615–620 (cit. on pp. 18, 80).

Arvanitopoulos, N. and S. Susstrunk (2014). “Seam carving for text line extraction on color and grayscale historical manuscripts”. In: *Frontiers in Handwriting Recognition, ICFHR*. IEEE, pp. 726–731 (cit. on p. 24).

Avidan, S. and A. Shamir (2007). “Seam Carving for Content-Aware Image Resizing”. In: *ACM Trans. Graph.* 26.3, p. 10 (cit. on pp. 24, 155).

Ba, J. L., J. R. Kiros, and G. E. Hinton (2016). “Layer Normalization”. In: *arXiv*. arXiv: 1607.06450 (cit. on pp. 54, 203).

Bäck, T. and H.-P. Schwefel (1993). “An Overview of Evolutionary Algorithms for Parameter Optimization”. In: *Evolutionary Computation* 1.1, pp. 1–23 (cit. on p. 56).

Badekas, E. and N. Papamarkos (2007). “Optimal combination of document binarization techniques using a self-organizing map neural network”. In: *Engineering Applications of Artificial Intelligence* 20.1, pp. 11–24 (cit. on pp. 19, 80).

Baechler, M. and R. Ingold (2010). “Medieval manuscript layout model”. In: *Document Engineering*, pp. 275–278 (cit. on p. 239).

Baechler, M., M. Liwicki, and R. Ingold (2013). “Text line extraction using DMLP classifiers for historical manuscripts”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1029–1033 (cit. on pp. 25, 29, 133, 141).

Baird, H. S. (2007). “The state of the art of document image degradation modelling”. In: *Digital Document Processing*. Springer, pp. 261–279 (cit. on pp. 66, 68).

Baldi, P., S. Brunak, P. Frasconi, G. Soda, and G. Pollastri (1999). “Exploiting the past and the future in protein secondary structure prediction.” In: *Bioinformatics (Oxford, England)* 15.11, pp. 937–946 (cit. on p. 36).

Baldi, P. and P. J. Sadowski (2013). “Understanding Dropout”. In: *Advances in Neural Information Processing Systems 26* 1, pp. 2814–2822 (cit. on p. 53).

Baronia, S. and A. Namboodiri (2013). “Ink-bleed reduction using layer separation”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 215–219 (cit. on p. 16).

Bar-Yosef, I., N. Hagbi, K. Kedem, and I. Dinstein (2009). “Line segmentation for degraded handwritten historical documents”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1161–1165 (cit. on p. 22).

Bauer, L. (1993). *Manual of Information to Accompany The Wellington Corpus of Written New Zealand English*. Tech. rep. Department of Linguistics, Victoria University, Wellington, New Zealand (cit. on p. 216).

Bengio, Y., R. Ducharme, P. Vincent, and C. Janvin (2003). “A Neural Probabilistic Language Model”. In: *The Journal of Machine Learning Research* 3, pp. 1137–1155 (cit. on p. 214).

Bengio, Y., J. Louradour, R. Collobert, and J. Weston (2009). “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48 (cit. on p. 54).

Bengio, Y., P. Simard, and P. Frasconi (1994). “Learning Long-Term Dependencies with Gradient Descent is Difficult”. In: *IEEE Transactions on Neural Networks* 5.2, pp. 157–166. arXiv: `arXiv:1211.5063v2` (cit. on pp. 49, 53).

Bergstra, J., R. Bardenet, Y. Bengio, and B. Kégl (2011). “Algorithms for Hyper-Parameter Optimization”. In: *Advances in Neural Information Processing Systems (NIPS)*, pp. 2546–2554 (cit. on p. 55).

Bergstra, J. and Y. Bengio (2012). “Random search for hyper-parameter optimization”. In: *The Journal of Machine Learning Research* 13, pp. 281–305 (cit. on p. 56).

Bernsen, J. (1986). “Dynamic thresholding of grey-level images”. In: *International conference on pattern recognition*, pp. 1251–1255 (cit. on p. 14).

Bertolami, R. and H. Bunke (2008). “Hidden Markov model-based ensemble methods for offline handwritten text line recognition”. In: *Pattern Recognition* 41.11, pp. 3452–3460 (cit. on p. 184).

Bertolami, R., S. Uchida, M. Zimmermann, and H. Bunke (2007). “Non-uniform slant correction for handwritten text line recognition”. In: *Internation Conferecence on Document Analysis and Recognition, ICDAR*. Vol. 1. IEEE, pp. 18–22 (cit. on p. 32).

Bi, M., Y. Qian, and K. Yu (2015). “Very deep convolutional neural networks for LVCSR”. In: *Sixteenth Annual Conference of the International Speech Communication Association* (cit. on p. 39).

Bianne-Bernard, A. L., F. Menasri, R. Al-Hajj Mohamad, C. Mokbel, C. Kermorvant, and L. Likforman-Sulem (2011). “Dynamic and contextual information in HMM modeling for handwritten word recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33.10, pp. 2066–2080 (cit. on pp. 33, 38, 184).

Biller, O., K. Kedem, I. Dinstein, and J. El-Sana (2012). “Evolution maps for connected components in text documents”. In: *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pp. 405–410 (cit. on p. 29).

Bilmes, J., K. Asanovic, C.-W. Chin, and J. Demmel (1997). “Using PHiPAC to speed error back-propagation learning”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing. ICASSP*. Vol. 5. IEEE, pp. 4153–4156 (cit. on p. 45).

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press (cit. on p. 212).

Bluche, T., H. Ney, and C. Kermorvant (2013a). “Feature Extraction with Convolutional Neural Networks for Handwritten Word Recognition”. In: *International Conference on Document Analysis and Recognition, ICDAR*, pp. 285–289 (cit. on p. 36).

Bluche, T., H. Ney, and C. Kermorvant (2014). “A Comparison of Sequence-Trained Deep Neural Networks and Recurrent Neural Networks Optical Modeling for Handwriting Recognition”. In: *International Conference on Statistical Language and Speech Processing*, pp. 1–12 (cit. on pp. 37, 183, 184, 206).

Bluche, T., H. Ney, and C. Kermorvant (2013b). “Tandem HMM with convolutional neural network for handwritten word recognition”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 2390–2394 (cit. on pp. 35, 37, 184).

Bluche, T., H. Ney, J. Louradour, and C. Kermorvant (2015). “Framewise and CTC training of Neural Networks for handwriting recognition”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 81–85 (cit. on p. 36).

Bosch, V., A. H. Toselli, and E. Vidal (2012). “Statistical Text Line Analysis in Handwritten Documents”. In: *Proc. Int. Conf. Frontiers in Handwriting Recognition*. Bari, Italy, pp. 201–206 (cit. on p. 27).

Bourlard, H. and N. Morgan (1994). *Connectionist speech recognition—A hybrid approach*. Vol. 247. Series in engineering and computer science. Kluwer Academic (cit. on p. 212).

Bozinovic, R. M. and S. N. Srihari (1989). “Off-Line Cursive Script Word Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.1, pp. 68–83 (cit. on pp. 30, 31).

Bradley, D. and G. Roth (2007). “Adaptive Thresholding using the Integral Image”. In: *Journal of Graphics, GPU, and Game Tools* 12.2, pp. 13–21 (cit. on p. 19).

Brink, A. D. (1992). “Thresholding of digital images using two-dimensional entropies”. In: *Pattern recognition* 25.8, pp. 803–808 (cit. on p. 13).

Brown, M. K. and S. Ganapathy (1983). “Preprocessing techniques for cursive script word recognition”. In: *Pattern recognition* 16.5, pp. 447–458 (cit. on pp. 30–32).

Bukhari, S. S., M. I. A. Al Azawi, F. Shafait, and T. M. Breuel (2010). “Document image segmentation using discriminative learning over connected components”. In: *Proceedings of the 8th IAPR International Workshop on Document Analysis Systems - DAS ’10*, pp. 183–190 (cit. on p. 143).

Bukhari, S. S., F. Shafait, and T. M. Breuel (2008). “Segmentation of curled textlines using active contours”. In: *International Workshop on Document Analysis Systems*, pp. 270–277 (cit. on p. 23).

Bukhari, S. S., F. Shafait, and T. M. Breuel (2009). “Script-independent handwritten textlines segmentation using active contours”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 446–450 (cit. on pp. 21, 23).

Bunke, H., M. Roth, and E. Schukat-Talamazzini (1995). “Off-line cursive handwriting recognition using hidden markov models”. In: *Pattern Recognition* 28.9, pp. 1399–1413 (cit. on p. 30).

Burr, D. J. (1982). “A normalizing transform for cursive script recognition”. In: *Int. Conf. Pattern Recognition*. Munich, pp. 1027–1030 (cit. on p. 31).

Buse, R., Z.-Q. Liu, and T. Caelli (1997). “A structural and relational approach to handwritten word recognition”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 27.5, pp. 847–861 (cit. on pp. 30, 32).

Caesar, T., J. M. Gloger, and E. Mandler (1995). “Estimating the baseline for written material”. In: *Proc. 3rd Int. Conf. Document Analysis and Recognition*. Vol. 1, pp. 382–385 (cit. on pp. 30, 31).

Caesar, T., J. M. Gloger, and E. Mandler (1993). “Preprocessing and feature extraction for a handwriting recognition system”. In: *Proceedings of the Document Analysis and Recognition*. IEEE, pp. 408–411 (cit. on p. 31).

Caillault, E., C. Viard-Gaudin, and A. R. Ahmad (2005). “MS-TDNN with global discriminant trainings”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*. Vol. 2005, pp. 856–860 (cit. on p. 36).

Canny, J. (1986). “A computational approach to edge detection.” In: *IEEE transactions on pattern analysis and machine intelligence* 8.6, pp. 679–698 (cit. on p. 106).

Cao, H. and V. Govindaraju (2009). “Preprocessing of low-quality handwritten documents using markov random fields”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.7, pp. 1184–1194 (cit. on p. 17).

Cao, Y., S. Wang, and H. Li (2003). “Skew detection and correction in document images based on straight-line fitting”. In: *Pattern Recognition Letters* 24.12, pp. 1871–1879 (cit. on p. 31).

Casey, R. G. (1970). “Moment normalization of handprinted characters”. In: *IBM Journal of Research and Development* 14.5, pp. 548–557 (cit. on p. 33).

Chang, Y. F., Y. T. Pai, and S. J. Ruan (2008). “An efficient thresholding algorithm for degraded document images based on intelligent block detection”. In: *Conference Proceedings - IEEE International Conference on Systems, Man and Cybernetics*, pp. 667–672 (cit. on p. 14).

Chen, K., M. Seuret, M. Liwicki, J. Hennebert, and R. Ingold (2015). “Page segmentation of historical document images with convolutional autoencoders”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1011–1015 (cit. on p. 143).

Chi, Z. and K. W. Wong (2001). “A two-stage binarization approach for document images”. In: *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing*, pp. 275–278 (cit. on p. 18).

Cho, K., B. van Merrienboer, D. Bahdanau, and Y. Bengio (2014). “On the Properties of Neural Machine Translation: Encoder–Decoder Approaches”. In: *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pp. 103–111 (cit. on p. 39).

Cho, K., B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio (2014). “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”. In: *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1724–1734 (cit. on p. 39).

Chorowski, J. K., D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio (2015). "Attention-Based Models for Speech Recognition". In: *Advances in Neural Information Processing Systems 28*, pp. 577–585 (cit. on p. 39).

Cireşan, D. C., U. Meier, and J. Schmidhuber (2012). "Multi-column Deep Neural Networks for Image Classification". In: *Computer Vision and Pattern Recognition (CVPR)*, pp. 3642–3649 (cit. on p. 205).

Cireşan, D., U. Meier, J. Masci, and J. Schmidhuber (2012). "Multi-column deep neural network for traffic sign classification". In: *Neural Networks* 32, pp. 333–338. arXiv: `arXiv:1202.2745v1` (cit. on p. 203).

Cohen, R., I. Dinstein, J. El-Sana, and K. Kedem (2014). "Using scale-space anisotropic smoothing for text line extraction in historical documents". In: *Image Analysis and Recognition*. Springer, pp. 349–358 (cit. on pp. 29, 141).

Costa, M. (1996). "Probabilistic interpretation of feedforward network outputs, with relationships to statistical prediction of ordinal quantities." In: *International journal of neural systems* 7.5, pp. 627–37 (cit. on p. 43).

Dahl, G. E., T. N. Sainath, and G. E. Hinton (2013). "Improving deep neural networks for LVCSR using rectified linear units and dropout". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 8609–8613 (cit. on pp. 38, 53).

Dahl, G. E., D. Yu, L. Deng, and A. Acero (2012). "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition". In: *IEEE Transactions on Audio, Speech and Language Processing* 20.1, pp. 30–42 (cit. on p. 38).

Dean, J., G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, and Q. V. Le (2012). "Large scale distributed deep networks". In: *Advances in Neural Information Processing Systems*, pp. 1223–1231 (cit. on p. 47).

Dembczynski, K. J., W. Waegeman, W. Cheng, and E. Hüllermeier (2011). “An exact algorithm for F-measure maximization”. In: *Advances in neural information processing systems*, pp. 1404–1412 (cit. on p. 188).

Deng, L., O. Abdel-Hamid, and D. Yu (2013). “A deep convolutional neural network using heterogeneous pooling for trading acoustic invariance with phonetic confusion”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 6669–6673 (cit. on p. 53).

Diem, M., F. Kleber, and R. Sablatnig (2013). “Text line detection for heterogeneous documents”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 743–747 (cit. on pp. 28, 141).

Doetsch, P., M. Kozielski, and H. Ney (2014). “Fast and Robust Training of Recurrent Neural Networks for Offline Handwriting Recognition”. In: *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*, pp. 279–284 (cit. on pp. 183, 184).

Dos Anjos, A. and H. R. Shahbazkia (2008). “BI-level image thresholding - A fast method”. In: *BIOSIGNALS 2008 - Proceedings of the 1st International Conference on Bio-inspired Systems and Signal Processing*. Vol. 2, pp. 70–76 (cit. on p. 14).

Dos Santos, R. P., G. S. Clemente, T. I. Ren, and G. D. C. Calvalcanti (2009). “Text line segmentation based on morphology and histogram projection”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 651–655 (cit. on p. 22).

Dreuw, P., P. Doetsch, C. Plahl, and H. Ney (2011). “Hierarchical hybrid MLP/HMM or rather MLP features for a discriminatively trained Gaussian HMM: A comparison for offline handwriting recognition”. In: *International Conference on Image Processing, ICIP*, pp. 3541–3544 (cit. on p. 37).

Dreuw, P., G. Heigold, and H. Ney (2009). “Confidence-based discriminative training for model adaptation in offline Arabic handwriting recognition”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 596–600 (cit. on p. 34).

Dreuw, P., G. Heigold, and H. Ney (2011). “Confidence and Margin-Based MMI/MPE Discriminative Training for Online Handwriting Recognition”. In: *International Journal of Document Analysis and Recognition* 14.3, pp. 273–288 (cit. on p. 184).

Drira, F. (2006). “Towards restoring historic documents degraded over time”. In: *Proceedings - Second International Conference on Document Image Analysis for Libraries, DIAL 2006*, pp. 350–357 (cit. on p. 16).

Duda, R. O., P. E. Hart, and D. G. Stork (2001). “Pattern classification. 2nd”. In: *Edition. New York* (cit. on p. 45).

Dzmitry Bahdana, D. Bahdanau, K. Cho, and Y. Bengio (2014). “Neural Machine Translation By Jointly Learning To Align and Translate”. In: *arXiv preprint*, pp. 1–15. arXiv: 1409.0473 (cit. on p. 39).

Easton, R. L., K. T. Knox, and W. A. Christens-Barry (2004). “Multispectral imaging of the Archimedes palimpsest”. In: *Proceedings - Applied Imagery Pattern Recognition Workshop*, pp. 111–116 (cit. on p. 16).

España Boquera, S. (2016). “Contributions to the joint segmentation and classification of sequences (My two cents on decoding and handwriting recognition)”. PhD thesis (cit. on p. 12).

Espana-Boquera, S., M. J. Castro-Bleda, J. Gorbe-Moya, and F. Zamora-Martinez (2011). “Improving Offline Handwritten Text Recognition with Hybrid HMM/ANN Models”. In: *IEEE Trans. on Pattern Analysis and Machine Intelligence* 33.4, pp. 767–779 (cit. on pp. 2, 6, 35, 36, 105, 150, 159, 161, 166, 184, 199, 211, 214–216, 224).

Farrahi Moghaddam, R. and M. Cheriet (2010). “A multi-scale framework for adaptive binarization of degraded document images”. In: *Pattern Recognition* 43.6, pp. 2186–2198 (cit. on p. 16).

Farrahi Moghaddam, R. and M. Cheriet (2012). “AdOtsu: An adaptive and parameterless generalization of Otsu’s method for document image binarization”. In: *Pattern Recognition* 45.6, pp. 2419–2431 (cit. on pp. 13–15).

Feldbach, M. and K. D. Tonnies (2001). “Line detection and segmentation in historical church registers”. In: *Intern. Conf. on Document Analysis and Recognition*, pp. 743–747 (cit. on pp. 21, 25).

Fernandez, D., J. Llados, A. Fornes, and R. Manmatha (2012). “On influence of line segmentation in efficient word segmentation in old manuscripts”. In: *International Conference on Frontiers in Handwriting Recognition, ICFHR*. IEEE, pp. 763–768 (cit. on p. 28).

Fernández-Mota, D., J. Lladós, and A. Fornés (2014). “A graph-based approach for segmenting touching lines in historical handwritten documents”. In: *International Journal on Document Analysis and Recognition* 17.3, pp. 293–312 (cit. on pp. 21, 28).

Fischer, A., M. Baechler, A. Garz, M. Liwicki, and R. Ingold (2014). “A Combined System for Text Line Extraction and Handwriting Recognition in Historical Documents”. In: *2014 11th IAPR International Workshop on Document Analysis Systems*, pp. 71–75 (cit. on pp. 228, 229).

Francis, W. N. and H. Kucera (1979). *Brown Corpus Manual, Manual of Information to accompany A Standard Corpus of Present-Day Edited American English*. Tech. rep. Department of Linguistics, Brown University, Providence, Rhode Island, US (cit. on p. 216).

Fung, C. C. and R. Chamchong (2010). “A review of evaluation of optimal binarization technique for character segmentation in historical manuscripts”. In: *3rd International Conference on Knowledge Discovery and Data Mining, WKDD 2010*, pp. 236–240 (cit. on p. 16).

Gao, Y., X. Ding, and C. Liu (2011). “A multi-scale text line segmentation method in freestyle handwritten documents”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 643–647 (cit. on p. 24).

Garz, A., A. Fischer, H. Bunke, and R. Ingold (2013). “A Binarization-Free Clustering Approach to Segment Curved Text Lines in Historical Manuscripts”. In: *International Conference Document Analysis and Recognition, ICDAR*. IEEE, pp. 1290–1294 (cit. on pp. 22, 140).

Garz, A., A. Fischer, R. Sablatnig, and H. Bunke (2012). “Binarization-free text line segmentation for historical documents based on interest point clustering”. In: *Intern. Wkshp. on Document Analysis Systems (DAS)*. IEEE, pp. 95–99 (cit. on pp. 24, 141).

Gatos, B., K. Ntirogiannis, and I. Pratikakis (2011). “DIBCO 2009: Document image binarization contest”. In: *International Journal on Document Analysis and Recognition* 14.1, pp. 35–44 (cit. on p. 13).

Gatos, B., K. Ntirogiannis, and I. Pratikakis (2009). “ICDAR 2009 Document Image Binarization Contest (DIBCO 2009).” In: *ICDAR*. Vol. 9, pp. 1375–1382 (cit. on pp. 217, 226, 237).

Gatos, B., N. Stamatopoulos, and G. Louloudis (2010). “ICFHR 2010 handwriting segmentation contest”. In: *International Conference on Frontiers in Handwriting Recognition, ICFHR*, pp. 737–742 (cit. on pp. 226, 237).

Gehring, J., M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin (2017). “Convolutional Sequence to Sequence Learning”. In: *arXiv preprint*. arXiv: 1705.03122 (cit. on p. 39).

Glorot, X. and Y. Bengio (2010). “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)* 9, pp. 249–256 (cit. on pp. 44, 203).

Goodfellow, I. J., D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio (2013). “Maxout Networks”. In: *arXiv preprint*, pp. 1319–1327. arXiv: 1302.4389 (cit. on p. 54).

Goodfellow, I., J. Pouget-Abadie, and M. Mirza (2014). “Generative Adversarial Networks”. In: *arXiv preprint*, pp. 1–9. arXiv: 1406.2661 (cit. on p. 54).

Gorbe-Moya, J., S. España-Boquera, F. Zamora-Martínez, and M. J. Castro-Bleda (2008). “Handwritten Text Normalization by using Local Extrema Classification”. In: *PRIS* 8, pp. 164–172 (cit. on pp. 31–33, 105, 106, 206, 211, 251).

Graves, A. (2008). “Supervised Sequence Labelling with Recurrent Neural Networks”. In: *Image Rochester NY*, p. 124 (cit. on p. 80).

Graves, A., S. Fernández, F. Gomez, and J. Schmidhuber (2006). “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks”. In: *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376 (cit. on p. 36).

Graves, A., S. Fernández, and J. Schmidhuber (2007). “Multi-Dimensional Recurrent Neural Networks”. In: *Proceedings of the International Conference on Artificial Neural Networks*. ICANN ’07. Springer-Verlag, pp. 549–558 (cit. on pp. 50, 77).

Graves, A., M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber (2009). “A Novel Connectionist System for Unconstrained Handwriting Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31.5, pp. 855–868 (cit. on pp. 34, 36, 184).

Graves, A., A.-r. Mohamed, and G. Hinton (2013). “Speech Recognition With Deep Recurrent Neural Networks”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP* 3, pp. 6645–6649. arXiv: `arXiv:1303.5778v1` (cit. on p. 50).

Graves, A. and J. Schmidhuber (2005). “Framewise phoneme classification with bidirectional LSTM networks”. In: *Proceedings of the International Joint Conference on Neural Networks*. Vol. 4, pp. 2047–2052 (cit. on p. 36).

Graves, A. and J. Schmidhuber (2009). “Offline handwriting recognition with multidimensional recurrent neural networks”. In: *Advances in Neural Information Processing Systems*, pp. 545–552 (cit. on pp. 33, 36).

Grézl, F., M. Karafiát, S. Kontár, and J. Černocký (2007). “Probabilistic and bottle-neck features for LVCSR of meetings”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*. Vol. 4 (cit. on p. 36).

Guerfali, W. and R. R. R. R. Plamondon (1993). “Normalizing and restoring on-line handwriting”. In: *Pattern Recognition* 26.3, pp. 419–431 (cit. on pp. 30, 31).

Gupta, M. D., S. Rajaram, N. Petrovic, and T. S. Huang (2006). “Models for patch based image restoration”. In: *Proceedings of the IEEE Computer So-*

ciety Conference on Computer Vision and Pattern Recognition. Vol. 2006 (cit. on p. 17).

Al-Haddad, L., C. W. Morris, and L. Boddy (2000). "Training radial basis function neural networks: effects of training set size and imbalanced training sets". In: *Journal of microbiological methods* 43.1, pp. 33–44 (cit. on pp. 64, 188).

El-Hajj, R., L. Likforman-Sulem, and C. Mokbel (2005). "Arabic handwriting recognition using baseline dependant features and hidden Markov modeling". In: *Internation Conferecence on Document Analysis and Recognition, ICDAR*. IEEE, pp. 893–897 (cit. on p. 38).

Hanasusanto, G. A., Z. Wu, and M. S. Brown (2010). "Ink-bleed reduction using functional minimization". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 825–832 (cit. on p. 16).

Hannun, A., C. Case, J. Casper, B. Catanzaro, G. Diamos, E. Elsen, R. Prenger, S. Satheesh, S. Sengupta, A. Coates, and A. Y. Ng (2014). "Deep Speech: Scaling up end-to-end speech recognition". In: *Arxiv*, pp. 1–12. arXiv: 1412.5567v2 (cit. on p. 38).

Hart, P. E., D. G. Stork, and J. Wiley (1973). "Pattern classification and scene analysis". In: *Artificial Intelligence* 4.2, pp. 139–143 (cit. on p. 20).

He, K., X. Zhang, S. Ren, and J. Sun (2015). "Deep Residual Learning for Image Recognition". In: *Arxiv.Org* 7.3, pp. 171–180. arXiv: 1512.03385 (cit. on pp. 54, 203).

Hedjam, R., R. F. Moghaddam, and M. Cheriet (2011). "A spatially adaptive statistical method for the binarization of historical manuscripts and degraded document images". In: *Pattern Recognition* 44.9, pp. 2184–2196 (cit. on p. 18).

Hedjam, R., H. Z. Nafchi, R. F. Moghaddam, M. Kalacska, and M. Cheriet (2015). "ICDAR 2015 Contest on MultiSpectral Text Extraction (MS-TEx 2015)". In: *International Conference Document Analysis and Recognition, ICDAR*. IEEE, pp. 1181–1185 (cit. on pp. 16, 200).

Hennig, A. and N. Sherkat (2002). “Exploiting zoning based on approximating splines in cursive script recognition”. In: *Pattern Recognition* 35.2, pp. 445–454 (cit. on p. 31).

Hidalgo, J. L., S. España, M. J. Castro, and J. A. Pérez (2005). “Enhancement and cleaning of handwritten data by using neural networks”. In: *Pattern Recognition and Image Analysis*. Springer, pp. 376–383 (cit. on pp. 18, 83).

Hinton, G. (2014). “Dropout : A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research (JMLR)* 15, pp. 1929–1958 (cit. on p. 52).

Hinton, G. E. and R. R. Salakhutdinov (2006). “Reducing the dimensionality of data with neural networks.” In: *Science (New York, N.Y.)* 313.5786, pp. 504–7 (cit. on p. 53).

Hinton, G. E., N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov (2012). “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv*, pp. 1–18. arXiv: 1207.0580 (cit. on p. 52).

Hinton, G., L. Deng, D. Yu, G. E. Dahl, A.-r. Mohamed, N. Jaitly, V. Vanhoucke, P. Nguyen, T. N. Sainath, B. Kingsbury, and A. Senior (2012). “Deep Neural Networks for Acoustic Modeling in Speech Recognition”. In: *Ieee Signal Processing Magazine* 29, pp. 82–97 (cit. on pp. 36, 38).

Hinton, G., O. Vinyals, and J. Dean (2015). “Distilling the knowledge in a neural network”. In: *arXiv preprint* (cit. on p. 201).

Hochreiter, S. (1998). “The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions”. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 06, pp. 107–116 (cit. on p. 49).

Hochreiter, S. and J. Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780 (cit. on pp. 36, 50).

Huang, Y., M. S. Brown, and D. Xu (2008). “A framework for reducing ink-bleed in old documents”. In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR* (cit. on p. 16).

Hull, J. (1998). “Document image skew detection: Survey and annotated bibliography”. In: *Series in Machine Perception and Artificial Intelligence* 29, pp. 40–64 (cit. on pp. 31, 149).

Ioffe, S. and C. Szegedy (2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *arXiv*, pp. 1–11. arXiv: 1502.03167 (cit. on pp. 54, 203).

Iosifidis, A., A. Tefas, and I. Pitas (2015). “DropELM: Fast neural network regularization with Dropout and DropConnect”. In: *Neurocomputing* 162, pp. 57–66 (cit. on p. 53).

Jaeger, S., S. Manke, J. Reichert, and A. Waibel (2001). “Online handwriting recognition: The NPen++ recognizer”. In: *International Journal on Document Analysis and Recognition* 3.3, pp. 169–180 (cit. on pp. 34, 36).

Jagroop Kaur, D. and R. Mahajan (2014). “A Review of Degraded Document Image Binarization Techniques”. In: *changes* 3.5 (cit. on p. 16).

Jansche, M. (2005). “Maximum expected F-measure training of logistic regression models”. In: *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, pp. 692–699 (cit. on p. 188).

Johansson, S., E. Atwell, R. Garside, and G. Leech (1986). *The Tagged LOB Corpus: User’s Manual*. Tech. rep. Norwegian Computing Centre for the Humanities, Bergen, Norway, p. 149 (cit. on pp. 216, 224).

Kang, L. and D. Doermann (2011). “Template based segmentation of touching components in handwritten text lines”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 569–573 (cit. on p. 26).

Kang, L., J. Kumar, P. Ye, and D. Doermann (2012). “Learning text-line segmentation using codebooks and graph partitioning”. In: *Frontiers in Handwriting Recognition (ICFHR), 2012 International Conference on*. IEEE, pp. 63–68 (cit. on p. 30).

Kavallieratou, E., N. Fakotakis, and G. Kokkinakis (1999). “Skew angle estimation in document processing using Cohen’s class distributions”. In: *Pattern Recognition Letters* 20.11-13, pp. 1305–1311 (cit. on p. 32).

Kavallieratou, E., N. Fakotakis, and G. Kokkinakis (2002). “Skew angle estimation for printed and handwritten documents using the Wigner-Ville distribution”. In: *Image and Vision Computing* 20.11, pp. 813–824 (cit. on p. 32).

Kennard, D. J. and W. A. Barrett (2006). “Separating lines of text in free-form handwritten historical documents”. In: *Proceedings - Second International Conference on Document Image Analysis for Libraries, DIAL 2006*. Vol. 2006, pp. 12–23 (cit. on p. 26).

Keysers, D., T. Deselaers, H. A. Rowley, L.-L. Wang, and V. Carbune (2016). “Multi-Language Online Handwriting Recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8828.c, pp. 1–1 (cit. on p. 34).

Khankasikam, K. (2013). “The Automatic Binarization Techniques Selection: An Artificial Neural Network Approach”. In: *International Journal of Digital Content Technology and its Applications* 7.9, p. 468 (cit. on p. 19).

Kieu, V. C., M. Visani, N. Journet, J.-P. Domenger, and R. Mullot (2012). “A character degradation model for grayscale ancient document images”. In: *International Conference on Pattern Recognition, ICPR*. IEEE, pp. 685–688 (cit. on p. 223).

Kieu, V. C., M. Visani, N. Journet, R. Mullot, and J. P. Domenger (2013). “An efficient parametrization of character degradation model for semi-synthetic image generation”. In: *Proceedings of the 2nd International Workshop on Historical Document Imaging and Processing*. ACM, pp. 29–35 (cit. on pp. 66, 68).

Kittler, J. and J. Illingworth (1985). “On threshold selection using clustering criteria”. In: *Systems, Man and Cybernetics, IEEE Transactions on* 5, pp. 652–655 (cit. on pp. 13, 14).

Kozielski, M., P. Doetsch, and H. Ney (2013). “Improvements in rwth’s system for off-line handwriting recognition”. In: *International Conference Docu-*

ment Analysis and Recognition, ICDAR. IEEE, pp. 935–939 (cit. on pp. 37, 183, 184, 206).

Kozielski, M., J. Forster, and H. Ney (2012). "Moment-based image normalization for handwritten text recognition". In: *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pp. 256–261 (cit. on pp. 33, 37, 38).

Krizhevsky, A., I. Sutskever, and G. E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., pp. 1097–1105 (cit. on pp. 49, 53, 170).

Kuk, J. G., N. I. Cho, and K. M. Lee (2008). "MAP-MRF approach for binarization of degraded document image". In: *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*. IEEE, pp. 2612–2615 (cit. on p. 17).

Kumar, J., W. Abd-Almageed, L. Kang, and D. Doermann (2010). "Handwritten arabic text line segmentation using affinity propagation". In: *Intern. Wkshp. on Document Analysis Systems*. ACM, pp. 135–142 (cit. on pp. 25, 26).

Kumar, J., L. Kang, D. Doermann, and W. Abd-Almageed (2011). "Segmentation of handwritten textlines in presence of touching components". In: *Document Analysis and Recognition (ICDAR), 2011 Intern. Conf. on*. IEEE, pp. 109–113 (cit. on pp. 25, 117).

Lecun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324 (cit. on pp. 48, 170).

Lelore, T. and F. Bouchara (2009). "Document image binarisation using markov field model". In: *International Conference Document Analysis and Recognition, ICDAR*. IEEE, pp. 551–555 (cit. on p. 17).

Lettner, M. and R. Sablatnig (2010). "Higher order MRF for foreground-background separation in multi-spectral images of historical manuscripts". In: *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*. ACM, pp. 317–324 (cit. on p. 17).

Li, J., X. Wang, and B. Xu (2013). “Understanding the dropout strategy and analyzing its effectiveness on LVCSR”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 7614–7618 (cit. on p. 53).

Li, S. Z. (2009). “Markov Random Field Modeling in Image Analysis”. In: *Security* i.4205, p. 371 (cit. on p. 17).

Li, Y., Y. Zheng, D. Doermann, and S. Jaeger (2008). “Script-independent text line segmentation in freestyle handwritten documents”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.8, pp. 1313–1329 (cit. on pp. 23, 121, 227, 228, 231).

Likforman-Sulem, L., A. Zahour, and B. Taconet (2007). “Text line segmentation of historical documents: a survey”. In: *International Journal of Document Analysis and Recognition, IJDAR* 9.2-4, pp. 123–138 (cit. on pp. 20, 62).

Long, J., E. Shelhamer, and T. Darrell (2014). “Fully convolutional networks for semantic segmentation”. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pp. 3431–3440 (cit. on pp. 59, 169).

Lopes, N. V., P. A. Mogadouro do Couto, H. Bustince, and P. Melo-Pinto (2010). “Automatic histogram threshold using fuzzy measures”. In: *IEEE Transactions on Image Processing* 19.1, pp. 199–204 (cit. on p. 14).

Louloudis, G., B. Gatos, I. Pratikakis, and C. Halatsis (2008). “Text line detection in handwritten documents”. In: *Pattern Recognition* 41.12, pp. 3758–3772 (cit. on p. 21).

Lu, L., X. Zhang, K. Cho, and S. Renals (2015). “A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition”. In: *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, pp. 3249–3253 (cit. on p. 39).

Lu, Z., R. Schwartz, and C. Raphael (2000). “Script-independent, HMM-based text line finding for OCR”. In: *Proc. 15th Int. Conf. Pattern Recognition*. Vol. 4, pp. 551–554 (cit. on pp. 27, 29).

Manmatha, R. and J. L. Rothfeder (2005). “A scale space approach for automatically segmenting words from historical handwritten documents”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.8, pp. 1212–1225 (cit. on p. 28).

Marcel, W., R. Ingold, and M. Liwicki (2016). “SDK Reinvented : Document Image Analysis Methods as RESTful Web Services”. In: *12th IAPR Workshop on Document Analysis Systems*, pp. 90–95 (cit. on pp. 201, 246).

Marinai, S., M. Gori, and G. Soda (2005). “Artificial neural networks for document analysis and recognition”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.1, pp. 23–35 (cit. on pp. 18, 72, 83).

Marti, U. V. and H. Bunke (2002a). “A full English sentence database for off-line handwriting recognition”. In: *International Journal on Document Analysis and Recognition* 5.1, pp. 39–46 (cit. on p. 224).

Marti, U. V. and H. Bunke (2002b). “Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition systems”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 15, pp. 65–90 (cit. on pp. 32, 33, 38).

Marukatat, S., T. Artières, P. Gallinari, B. Dorizzi, T. Artieres, R. Gallinari, and B. Dorizzi (2001). “Sentence recognition through hybrid neuro-Markovian modeling”. In: *Proceedings of Sixth International Conference on Document Analysis and Recognition*, pp. 731–735 (cit. on pp. 35, 36).

Mehmet, S. and S. Bulent (2004). “Survey over image thresholding techniques and quantitative performance evaluation”. In: *Journal of Electronic Imaging* 13, p. 220 (cit. on p. 13).

Mehrara, H., M. Zahedinejad, and A. Pourmohammad (2009). “Novel edge detection using BP neural network based on threshold binarization”. In: *International Conference on Computer and Electrical Engineering, ICCEE*. Vol. 2. IEEE, pp. 408–412 (cit. on p. 18).

Miyoshi, T., T. Nagasaki, and H. Shinjo (2009). “Character normalization methods using moments of gradient features and normalization cooperated feature extraction”. In: *Chinese Conference on Pattern Recognition, CCPR*. IEEE, pp. 1–5 (cit. on p. 33).

Mohamed, A. R., G. Hinton, and G. Penn (2012). “Understanding how deep belief networks perform acoustic modelling”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 4273–4276 (cit. on p. 38).

Morita, M., F. Bortolozzi, J. Facon, S. Garnés, and R. Sabourin (1999). “Mathematical morphology and weighted least squares to correct handwriting baseline skew”. In: *International Conference on Document Analysis and Recognition, ICDAR*, pp. 430–434 (cit. on p. 31).

Moysset, B., C. Kermorvant, C. Wolf, and J. Louradour (2015). “Paragraph text segmentation into lines with Recurrent Neural Networks”. In: *Internation Conferecence on Document Analysis and Recognition, ICDAR*. IEEE, pp. 456–460 (cit. on pp. 27, 29).

Munkres, J. (1957). “Algorithms for the Assignment and Transportation Problems”. In: *Journal of the Society for Industrial and Applied Mathematics* 5.1, pp. 32–38 (cit. on p. 121).

Musicant, D. R., V. Kumar, A. Ozgur, et al. (2003). “Optimizing F-Measure with Support Vector Machines.” In: *FLAIRS Conference*, pp. 356–360 (cit. on p. 188).

Nagy, G. (2000). “Twenty years of document image analysis in PAMI”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1, pp. 38–62 (cit. on pp. 12, 20).

Nelder, J. a. and R. Mead (1964). “A simplex method for function minimization”. In: *The Computer Journal* 7.4, pp. 308–313 (cit. on p. 82).

Niblack, W. (1985). *An introduction to digital image processing*. Strandberg Publishing Company (cit. on pp. 13, 14).

Nicolaou, A. and B. Gatos (2009). “Handwritten text line segmentation by shredding text into its lines”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 626–630 (cit. on p. 23).

Nicolaou, A., R. Ingold, and M. Liwicki (2014). “Binarization with the Local Otsu Filter: Integral histograms for document image analysis”. In: *Lecture Notes in Computer Science* 8746, pp. 176–190 (cit. on p. 19).

Nikolaou, N., M. Makridis, B. Gatos, N. Stamatopoulos, and N. Papamarkos (2010). “Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths”. In: *Image and Vision Computing* 28.4, pp. 590–604 (cit. on p. 23).

Nina, O., B. Morse, and W. Barrett (2011). “A recursive otsu thresholding method for scanned document binarization”. In: *IEEE Workshop on Applications of Computer Vision, WACV*, pp. 307–314 (cit. on p. 16).

Ntirogiannis, K., B. Gatos, and I. Pratikakis (2014a). “A combined approach for the binarization of handwritten document images”. In: *Pattern Recognition Letters* 35.1, pp. 3–15 (cit. on pp. 14, 15).

Ntirogiannis, K., B. Gatos, and I. Pratikakis (2008). “An objective evaluation methodology for document image binarization techniques”. In: *International Workshop on Document Analysis Systems, DAS*, pp. 217–224 (cit. on p. 19).

Ntirogiannis, K., B. B. Gatos, and I. Pratikakis (2013). “Performance evaluation methodology for historical document image binarization”. In: *Image Processing, IEEE Transactions on* 22.2, pp. 595–609 (cit. on pp. 19, 69, 72).

Ntirogiannis, K., B. Gatos, and I. Pratikakis (2014b). “ICFHR2014 Competition on Handwritten Document Image Binarization (H-DIBCO 2014)”. In: *International Conference on Frontiers in Handwriting Recognition, ICFHR*. IEEE, pp. 809–813 (cit. on p. 13).

Och, F. J. (2003). “Minimum Error Rate Training in Statistical Machine Translation”. In: *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*. Vol. 1001, pp. 160–167 (cit. on p. 216).

Otsu, N. (1975). “A threshold selection method from gray-level histograms”. In: *Automatica* 11.1, pp. 23–27 (cit. on pp. 13, 14, 62).

Ouwayed, N., A. Belaïd, and F. Auger (2010). “General text line extraction approach based on locally orientation estimation”. In: *Document Recognition and Retrieval Conference*. Intern. Society for Optics and Photonics, pp. 1–10 (cit. on pp. 21, 124).

Öztop, E., A. Y. Mülayim, V. Atalay, and F. Yarman-Vural (1999). “Repulsive attractive network for baseline extraction on document images”. In: *Signal Processing* 75.1, pp. 1–10 (cit. on p. 20).

Pai, Y.-T., Y.-F. Chang, and S.-J. Ruan (2010). “Adaptive thresholding algorithm: Efficient computation technique based on intelligent block detection for degraded document images”. In: *Pattern Recognition* 43.9, pp. 3177–3187 (cit. on pp. 14, 19).

Papavassiliou, V., V. Katsouros, and G. Carayannis (2010). “A morphological approach for text-line segmentation in handwritten documents”. In: *International Conference on Frontiers in Handwriting Recognition, ICFHR*, pp. 19–24 (cit. on p. 23).

Papavassiliou, V., T. Stafylakis, V. Katsouros, and G. Carayannis (2010). “Handwritten document image segmentation into text lines and words”. In: *Pattern Recognition* 43.1, pp. 369–377 (cit. on p. 22).

Paredes, R., E. Kavallieratou, and R. D. Lins (2010). “ICFHR 2010 contest: Quantitative evaluation of binarization algorithms”. In: *International*. IEEE, pp. 733–736 (cit. on p. 72).

Pastor, M., A. Toselli, and E. Vidal (2004). “Projection profile based algorithm for slant removal”. In: *Proceedings of the 2004 International Conference on Image Analysis and Recognition (ICIAR04)* (cit. on pp. 31, 32).

Pastor-Pellicer, J., M. J. Castro-Bleda, and J. L. Adelantado-Torres (2015). “esCam: A Mobile Application to Capture and Enhance Text Images”. In: *Advances in Computational Intelligence*. Springer International Publishing, pp. 601–604 (cit. on p. 19).

Pastor-Pellicer, J., S. España-Boquera, F. Zamora-Martínez, M. Zeshan Afzal, and M. J. Castro-Bleda (2015). “Insights on the use of convolutional neural networks for document image binarization”. In: *International Work-*

Conference on Artificial Neural Networks, IWANN. Vol. 9095, pp. 115–126 (cit. on pp. 98, 215).

Pastor-Pellicer, J., M. Z. Afzal, M. Liwicki, and M. J. Castro-Bleda (2016). "Complete System for Text Line Extraction Using Convolutional Neural Networks and Watershed Transform". In: *International Workshop on Document Analysis Systems, DAS*, pp. 30–35 (cit. on pp. 31, 146).

Pastor-Pellicer, J., S. España-Boquera, M. J. Castro-Bleda, and F. Zamora-Martínez (2015). "A combined Convolutional Neural Network and Dynamic Programming approach for text line normalization". In: *Internation Conferecence on Document Analysis and Recognition, ICDAR* (cit. on pp. 25, 163, 169, 176).

Pastor-Pellicer, J., S. España-Boquera, P. Zamora-Martínez, and M. Castro-Bleda (2014). "Handwriting Normalization by Zone Estimation Using HMM/ANNs". In: *International Conference on Frontiers in Handwriting Recognition*, pp. 633–638 (cit. on pp. 163, 224).

Pastor-Pellicer, J., A. Garz, R. Ingold, and M. J. Castro-Bleda (2015). "Combining Learned Script Points and Combinatorial Optimization for Text Line Extraction". In: *3rd International Workshop on Historical Document Image and Processing* (cit. on pp. 121, 127, 145).

Pastor-Pellicer, J., F. Zamora-Martínez, S. España-Boquera, and M. J. Castro-Bleda (2013). "F-measure as the error function to train neural networks". In: *International Work-Conference on Artificial Neural Networks, IWANN*. Vol. 7902 LNCS. PART 1, pp. 376–384 (cit. on pp. 64, 195).

Pham, V., T. Bluche, C. Kermorvant, and J. Louradour (2014). "Dropout Improves Recurrent Neural Networks for Handwriting Recognition". In: *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, pp. 285–290 (cit. on pp. 53, 183).

Phillips, I. T. and A. K. Chhabra (1999). "Empirical performance evaluation of graphics recognition systems". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21.9, pp. 849–870 (cit. on p. 226).

Plamondon, R. and S. N. Srihari (2000). “On-Line and Off-Line Handwriting Recognition : A Comprehensive Survey”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.1, pp. 63–84 (cit. on p. 30).

Pletschacher, S. and A. Antonacopoulos (2010). “The PAGE (Page Analysis and Ground-truth Elements) format framework”. In: *Proceedings - International Conference on Pattern Recognition*, pp. 257–260 (cit. on p. 235).

Plötz, T. and G. A. Fink (2009). “Markov models for offline handwriting recognition: a survey”. In: *International Journal of Document Analysis and Recognition, IJDAR* 12, pp. 269–298 (cit. on p. 184).

Powalka, R. K., N. Sherkat, and R. J. Whitrow (1994). “Feature Extraction: On the Importance of Zoning Information in Cursive Script Recognition”. In: *Progress in Image Analysis and Processing*, p. 342 (cit. on p. 31).

Poznanski, A. and L. Wolf (2016). “CNN-N-Gram for Handwriting Word Recognition”. In: *Computer Vision and Pattern Recognition, CVPR*, pp. 2305–2314 (cit. on pp. 183, 184).

Pratikakis, I., B. Gatos, and K. Ntirogiannis (2012). “ICFHR 2012 Competition on Handwritten Document Image Binarization (H-DIBCO 2012).” In: *ICFHR* 12, pp. 18–20 (cit. on pp. 13, 217).

Pratikakis, I., B. Gatos, and K. Ntirogiannis (2010). “H-DIBCO 2010 - Handwritten document image binarization competition”. In: *Proceedings - 12th International Conference on Frontiers in Handwriting Recognition, ICFHR 2010*, pp. 727–732 (cit. on pp. 13, 192, 217).

Pratikakis, I., B. Gatos, and K. Ntirogiannis (2011). “ICDAR 2011 Document Image Binarization Contest (DIBCO 2011)”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1506–1510 (cit. on p. 13).

Pratikakis, I., B. Gatos, and K. Ntirogiannis (2013). “ICDAR 2013 document image binarization contest (DIBCO 2013)”. In: *Internation Conferecence on Document Analysis and Recognition, ICDAR*. IEEE, pp. 1471–1476 (cit. on pp. 13, 89, 217).

Pun, T. (1980). “A new method for grey-level picture thresholding using the entropy of the histogram”. In: *Signal Processing* 2.3, pp. 223–237 (cit. on p. 14).

Rabaev, I., O. Biller, J. El-Sana, K. Kedem, and I. Dinstein (2013). “Text line detection in corrupted and damaged historical manuscripts”. In: *Internation Conferecence on Document Analysis and Recognition, ICDAR*. IEEE, pp. 812–816 (cit. on pp. 29, 141).

Rabiner, L. R. (1989). *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition* (cit. on pp. 35, 212).

Rabiner, L. and B. H. Huang (1993). *Fundamentals of Speech Recognition*. Prentice-Hall (cit. on p. 152).

Rangoni, Y., F. Shafait, and T. Breuel (2009). “OCR Based Thresholding.” In: *11th IAPR Conference on Machine Vision Applications Vision Applications*, pp. 3–6 (cit. on p. 19).

Rani, J. and D. Parkash (2015). *Degraded Document Image Binarization Techniques* (cit. on p. 16).

Razak, Z., K. Zulkiflee, and M. Idris (2008). “Off-line handwriting text line segmentation: a review”. In: *International journal of computer science and network security* 8.7, pp. 12–20 (cit. on p. 21).

Rohini, S. and S. Mohanavel (2012). “Segmentation of touching, overlapping, skewed and short handwritten text lines”. In: *International Journal of Computer Applications* 49.19 (cit. on p. 26).

Romero, V., M. Pastor, A. H. Toselli, and E. Vidal (2006). “Criteria for handwritten off-line text size normalization”. In: *Intern. Conf. on Visualization, Imaging, and Image Processing* (cit. on p. 21).

Romero, V., A. Fornés, N. Serrano, J. A. Sánchez, A. H. Toselli, V. Frinken, E. Vidal, and J. Lladós (2013). “The ESPOSALLES database: An ancient marriage license corpus for off-line handwriting recognition”. In: *Pattern Recognition* 46.6, pp. 1658–1669 (cit. on p. 200).

Rosenblatt, F. (1958). “The perceptron: a probabilistic model for information storage and organization in the brain”. In: *Psychol Rev.* 65.6, pp. 386–408 (cit. on p. 42).

Roy, S., P. Shivakumara, P. P. Roy, and C. L. Tan (2012). “Wavelet-Gradient-Fusion for Video Text Binarization”. In: *International Conference on Pattern Recognition* Icpr, pp. 3300–3303 (cit. on p. 16).

Russakovsky, O., J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei (2015). “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3, pp. 211–252 (cit. on p. 170).

Saabni, R. and J. El-Sana (2011). “Language-independent text lines extraction using seam carving”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 563–568 (cit. on p. 24).

Sahoo, P. K., S. Soltani, and A. K. C. Wong (1988). *A survey of thresholding techniques* (cit. on p. 13).

Sahoo, P., C. Wilkins, and J. Yeager (1997). “Threshold selection using Renyi’s entropy”. In: *Pattern Recognition* 30.1, pp. 71–84 (cit. on p. 14).

Sainath, T. N., B. Kingsbury, and B. Ramabhadran (2012). “Auto-encoder bottleneck features using deep belief networks”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 4153–4156 (cit. on p. 36).

Sainath, T. N., A. R. Mohamed, B. Kingsbury, and B. Ramabhadran (2013). “Deep convolutional neural networks for LVCSR”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 8614–8618 (cit. on p. 39).

Sankur, B. and M. Sezgin (2001). “Image thresholding techniques: A survey over categories”. In: *Pattern Recognition*, pp. 1–35 (cit. on p. 13).

Saon, G., H.-K. J. Kuo, S. Rennie, and M. Picheny (2015). “The IBM 2015 English Conversational Telephone Speech Recognition System”. In: *Interspeech*, pp. 3–7 (cit. on p. 39).

Sauvola, J. and M. Pietikäinen (2000). “Adaptive document image binarization”. In: *Pattern Recognition* 33.2, pp. 225–236 (cit. on pp. 13, 14, 62).

Sayre, K. M. (1973). “Machine recognition of handwritten words: A project report”. In: *Pattern Recognition* 5.3, pp. 213–228 (cit. on p. 35).

Schenk, J. and G. Rigoll (2006). “Novel hybrid NN/HMM modelling techniques for on-line handwriting recognition”. In: *Tenth International Workshop on Frontiers in Handwriting Recognition*. Suvisoft (cit. on p. 36).

Schenkel, M., I. Guyon, and D. Henderson (1995). “On-line cursive script recognition using time-delay neural networks and hidden Markov models”. In: *Machine Vision and Applications* 8.4, pp. 215–223 (cit. on p. 36).

Schuster, M. and K. K. Paliwal (1997). “Bidirectional recurrent neural networks”. In: *IEEE Transactions on Signal Processing* 45.11, pp. 2673–2681 (cit. on pp. 36, 50).

Schwenk, H. (2007). “Continuous space language models”. In: *Computer Speech and Language* 21.3, pp. 492–518 (cit. on p. 214).

Seide, F., G. Li, X. Chen, and D. Yu (2011). “Feature engineering in Context-Dependent Deep Neural Networks for conversational speech transcription”. In: *IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU*, pp. 24–29 (cit. on p. 38).

Seiler, R., M. Schenkel, and F. Eggimann (1996). “Off-line Cursive Handwriting Recognition Compared with On-line Recognition”. In: *Proc. 13th Int. Conf. on Pattern Recognition*, pp. 505–509 (cit. on p. 32).

Seltzer, M. L., D. Yu, and Y. Wang (2013). “An investigation of deep neural networks for noise robust speech recognition”. In: *International Conference on Acoustics, Speech and Signal Processing, ICASSP*, pp. 7398–7402 (cit. on p. 53).

Senior, A. W. and A. J. Robinson (1998). “An off-line cursive handwriting recognition system”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.3, pp. 309–321 (cit. on pp. 35, 36).

Sercu, T., C. Puhrsch, B. Kingsbury, and Y. LeCun (2015). “Very Deep Multilingual Convolutional Neural Networks for LVCSR”. In: *arXiv*, pp. 2–6. arXiv: 1509.08967 (cit. on p. 39).

Sermanet, P., S. Chintala, and Y. LeCun (2012). “Convolutional neural networks applied to house numbers digit classification”. In: *International Conference on Pattern Recognition (ICPR)*, pp. 3288–3291 (cit. on p. 49).

Shafait, F., D. Keysers, and T. M. Breuel (2006). “Pixel-accurate representation and evaluation of page segmentation in document images”. In: *International Conference on Pattern Recognition, ICPR*. Vol. 1, pp. 872–875 (cit. on p. 102).

Shafait, F., D. Keysers, and T. Breuel (2008). “Efficient implementation of local adaptive thresholding techniques using integral images”. In: *SPIE Document Imaging and Retrieval*, pp. 1–5 (cit. on p. 19).

Shi, Z. and V. Govindaraju (1997). “Segmentation and recognition of connected handwritten numeral strings”. In: *Pattern Recognition* 30.9, pp. 1501–1504 (cit. on p. 23).

Shi, Z., S. Setlur, and V. Govindaraju (2009). “A steerable directional local profile technique for extraction of handwritten Arabic text lines”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 176–180 (cit. on p. 23).

Shippert, P. (2003). “Introduction to hyperspectral image analysis”. In: *Online Journal of Space Communication* 3 (cit. on p. 16).

Simonyan, K. and A. Zisserman (2014). “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *ArXiv*, pp. 1–14. arXiv: 1409.1556 (cit. on pp. 39, 170).

Singer, E. and R. P. Lippman (1992). “A speech recognizer using radial basis function neural networks in an HMM framework”. In: *International Conference on Acoustics, Speech and Signal Processing, ICASSP*. Vol. 1, pp. 629–632 (cit. on p. 35).

Slavík, P. and V. Govindaraju (2001). “Equivalence of different methods for slant and skew corrections in word recognition applications”. In: *IEEE*

Transactions on Pattern Analysis and Machine Intelligence 23.3, pp. 323–326 (cit. on p. 32).

Snelson, E. and Z. Ghahramani (2006). “Sparse Gaussian Processes using Pseudo-inputs”. In: *Advances in Neural Information Processing Systems 18*, pp. 1257–1264 (cit. on p. 56).

Snoek, J., H. Larochelle, and R. P. Adams (2012). “Practical Bayesian Optimization of Machine Learning Algorithms”. In: *Adv. Neural Inf. Process. Syst. 25*, pp. 1–9 (cit. on p. 56).

Srivastava, N. (2013). “Improving neural networks with dropout”. PhD thesis (cit. on p. 53).

Stadermann, J. and G. Rigoll (2004). “A hybrid SVM/HMM acoustic modeling approach to automatic speech recognition”. In: *margin* 10, p. 1 (cit. on p. 35).

Stafylakis, T., V. Papavassiliou, V. Katsouros, and G. Carayannis (2008). “Robust text-line and word segmentation for handwritten documents images”. In: *International Conference on Acoustics Speech and Signal Processing, ICASSP*, pp. 3393–3396 (cit. on p. 29).

Stamatopoulos, N., B. Gatos, G. Louloudis, U. Pal, and A. Alaei (2013). “ICDAR 2013 handwriting segmentation contest”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1402–1406 (cit. on pp. 200, 226, 237, 256).

Stolcke, A. (2002). “SRILM: an extensible language modeling toolkit”. In: *Proc. Int. Conf. on Spoken Language Processing, ICSLP*, pp. 901–904 (cit. on p. 216).

Su, B., S. Lu, and C. L. Tan (2010). “A self-training learning document binarization framework”. In: *International Conference on Pattern Recognition, ICPR*. IEEE, pp. 3187–3190 (cit. on p. 18).

Su, B., S. Lu, and C. L. Tan (2011). “Combination of document image binarization techniques”. In: *International Conference Document Analysis and Recognition, ICDAR*. IEEE, pp. 22–26 (cit. on pp. 18, 80).

Su, B., S. Lu, and C. L. Tan (2013). “Robust document image binarization technique for degraded document images”. In: *IEEE Transactions on Image Processing* 22.4, pp. 1408–1417 (cit. on p. 16).

Su, F. and A. Mohammad-Djafari (2007). “Bayesian separation of document images with hidden Markov model”. In: *arXiv preprint arXiv:0705.2461* (cit. on p. 18).

Sun, C. and D. Si (1997). “Skew and slant correction for document images using gradient direction”. In: *Proceedings of the Fourth International Conference on Document Analysis and Recognition*. Vol. 1, pp. 142–146 (cit. on p. 31).

Sutskever, I., O. Vinyals, and Q. V. Le (2014). “Sequence to Sequence Learning with Neural Networks”. In: *Conference on Neural Information Processing Systems, NIPS*, p. 9 (cit. on p. 39).

Szegedy, C., W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich (2014). “Going Deeper with Convolutions”. In: *arXiv preprint*, pp. 1–12. arXiv: 1409.4842 (cit. on pp. 170, 203).

Toselli, A. H., A. Juan, J. González, I. Salvador, E. Vidal, F. Casacuberta, D. Keysers, and H. Ney (2004). “Integrated Handwriting Recognition and Interpretation using Finite-State Models”. In: *International Journal of Pattern Recognition and Artificial Intelligence* 18.4, pp. 519–539 (cit. on pp. 31, 33, 36, 38, 166, 171, 214).

Toselli, A. H., V. Romero, M. Pastor, and E. Vidal (2010). “Multimodal interactive transcription of text images”. In: *Pattern Recognition* 43.5, pp. 1814–1825 (cit. on p. 184).

Trier, O. and T. Taxt (1995). “Evaluation of binarization methods for document images”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 17.3, pp. 312–315 (cit. on p. 69).

Valizadeh, M. and E. Kabir (2012). “Binarization of degraded document image based on feature space partitioning and classification”. In: *International Journal on Document Analysis and Recognition, IJDAR* 15.1, pp. 57–69 (cit. on p. 16).

Varga, T. and H. Bunke (2003). “Generation of synthetic training data for an HMM-based handwriting recognition system”. In: *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 618–622 (cit. on pp. 66, 68).

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin (2017). “Attention Is All You Need”. In: *ArXiv e-prints*. arXiv: 1706.03762 [cs.CL] (cit. on p. 39).

Veselý, K., A. Ghoshal, L. Burget, and D. Povey (2013). “Sequence-discriminative training of deep neural networks”. In: *Interspeech*, pp. 2345–2349 (cit. on p. 38).

Vilar, J. M. (2008). “Efficient computation of confidence intervals for word error rates”. In: *ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings*, pp. 5101–5104 (cit. on p. 176).

Vincent, L. and P. Soille (1991). “Watersheds in digital spaces: an efficient algorithm based on immersion simulations”. In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6, pp. 583–598 (cit. on p. 136).

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010). “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *Journal of Machine Learning Research* 11.3, pp. 3371–3408 (cit. on pp. 53, 169).

Vinciarelli, A. and J. Luettin (2001). “A new normalization technique for cursive handwritten words”. In: *Pattern Recognition Letters* 22.9, pp. 1043–1050 (cit. on pp. 30–32).

Wan, L., M. Zeiler, S. Zhang, Y. LeCun, and R. Fergus (2013). “Regularization of neural networks using dropconnect”. In: *Icml* 1, pp. 109–111 (cit. on pp. 53, 203).

Wang, C., N. Komodakis, and N. Paragios (2013). “Markov Random Field modeling, inference & learning in computer vision & image understanding: A survey”. In: *Computer Vision and Image Understanding* 117.11, pp. 1610–1627 (cit. on p. 17).

Wang, S. I. and C. D. Manning (2013). “Fast dropout training”. In: *Proceedings of the 30th International Conference on Machine Learning* 28, pp. 118–126 (cit. on p. 53).

Werbos, P. J. (1990). “Backpropagation Through Time: What It Does and How to Do It”. In: *Proceedings of the IEEE* 78.10, pp. 1550–1560 (cit. on p. 50).

Williams, R. J. and J. Peng (1990). “An efficient gradient-based algorithm for on-line training of recurrent network trajectories”. In: *Neural Comput.* 2.4, pp. 490–501 (cit. on p. 50).

Wolf, C. (2010). “Document Ink Bleed-Through Removal with Two Hidden Markov Random Fields and a Single Observation Field”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 32.3, pp. 431–447 (cit. on pp. 16, 17).

Wolf, C. and D. Doermann (2002). “Binarization of low quality text using a markov random field model”. In: *16th International Conference on Pattern Recognition, 2002. Proceedings*. Vol. 3. IEEE, pp. 160–163 (cit. on p. 17).

Wong, A. and P. Sahoo (1989). “A gray-level threshold selection method based on maximum entropy principle”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 19.4, pp. 866–871 (cit. on p. 14).

Wong, K. Y., R. G. Casey, and F. M. Wahl. (1982). “Document Analysis system”. In: *IBM Journal of Research and Developement* 26.6, pp. 647–655 (cit. on p. 20).

Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean (2016). “Google’s Neural Machine Translation System: Bridging the Gap between Human and Machine Translation”. In: *ArXiv e-prints*, pp. 1–23. arXiv: `1609.08144v2` (cit. on p. 39).

Wu, Y., S. Rawls, W. AbdAlmageed, and P. Natarajan (2015). “Learning document image binarization from data”. In: *arXiv preprint*. arXiv: `1505.00529` (cit. on p. 18).

Yanikoglu, B. a. and L. Vincent (1998). “Pink Panther: a Complete Environment for Ground-Truthing and Benchmarking Document Page Segmentation”. In: *Pattern Recognition* 31.9, pp. 1191–1204 (cit. on p. 248).

Yin, F. and L. Cheng-Lin (2007). “Handwritten text line extraction based on minimum spanning tree clustering”. In: *2007 International Conference on Wavelet Analysis and Pattern Recognition* 3 (cit. on p. 21).

Yin, F. and C. L. Liu (2009). “Handwritten Chinese text line segmentation by clustering with distance metric learning”. In: *Pattern Recognition* 42.12, pp. 3146–3157 (cit. on p. 21).

Zamora-Martínez, F., V. Frinken, S. España-Boquera, M. J. Castro-Bleda, A. Fischer, and H. Bunke (2014). “Neural network language models for off-line handwriting recognition”. In: *Pattern Recognition* 47.4, pp. 1642–1652 (cit. on pp. 148, 184, 216, 224).

Zamora-Martínez, F. (2012). “Contributions to Connectionist Language Modeling and its application to Sequence Recognition and Machine Translation”. PhD thesis. Universitat Politècnica de València (cit. on p. 215).

Zamora-Martínez, F., S. España-Boquera, J. Gorbe-Moya, J. Pastor-Pellicer, and A. Palacios-Corella (2013). *APRIL-ANN toolkit, A Pattern Recognizer In Lua with Artificial Neural Networks* (cit. on pp. 8, 209).

Zeiler, M. D. (2012). “ADADELTA: an adaptive learning rate method”. In: *arXiv preprint*. arXiv: 1212.5701 (cit. on p. 161).

Zeiler, M. D. and R. Fergus (2013). “Stochastic Pooling for Regularization of Deep Convolutional Neural Networks”. In: *International Conference on Representation Learning*, pp. 1–9 (cit. on p. 49).

Zhou, Z.-H. and X.-Y. Liu (2006). “Training cost-sensitive neural networks with methods addressing the class imbalance problem”. In: *IEEE Transactions on Knowledge and Data Engineering* 18.1, pp. 63–77 (cit. on p. 188).

Zhou, Z., L. Li, and C. L. Tan (2010). “Edge based binarization for video text images”. In: *International Conference on Pattern Recognition, ICPR*, pp. 133–136 (cit. on p. 16).

Zi, G. and D. Doermann (2004). “Document Image Ground Truth Generation from Electronic Text”. In: *International Conference on Pattern Recognition*. Vol. 2, pp. 1–4 (cit. on pp. 66, 68).