UNIVERSITAT POLITÈCNICA DE VALÈNCIA

MIARFID Official Master's Degree in Artificial Intelligence, Pattern Recognition and Digital Imaging

Departamento de Sistemas Informáticos y Computación
Universitat Politècnica de València

Análisis de Maquetación ("Layout") en imágenes de texto manuscrito mediante Redes Neuronales

Trabajo fin de Máster

Máster Universitario en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital

Autor: José Ramón Prieto Fontcuberta
Tutor: Enrique Vidal Ruiz
Cotutor: Alejandro H. Toselli
Director experimental: Lorenzo Quirós Diaz

Curso 2018-2019

Agradecimientos

A Enrique, Alejandro y Lorenzo por la paciencia y correcciones en este trabajo.

A Tatiana por todo el apoyo.

Resum

Existeixen grans col·leccions de manuscrits, els quals contenen informació molt valuosa sobre aspectes crucials de la història de la nostra societat. Existeix tal quantitat de documents que de forma manual trigaríem anys, o inclús segles, en poder extraure tota la informació, on la majoria és textual. A causa d'això, es tracta d'utilitzar tècniques de maquetació i reconeixement de text manuscrit de les imatges de forma automàtica a fi de poder comprendre millor, i de manera més eficient, la informació que proporcionen aquestes col·leccions.

Aquest Treball de Fi de Màster s'ha centrat en el desenvolupament i avaluació de diferents tècniques d'aprenentatge profund per a realitzar la maquetació de pàgines amb alt valor històric.

Pel que aquest treball gira en torna a dues tasques. La primera, la segmentació de zones en un corpus del segle XIV al segle XIX. Dit corpus està compost majorment per taules, habilitant un posterior anàlisi per permetre realitzar consultes estructurades. La segona tasca tracta de la separació de registres en una col·lecció del segle XIV al segle XV dictats pel rei de França. Dita separació ajudaria a la recerca de temes concrets de l'època, així com possibles sentències escrites en els dits registres.

A més, s'ha tractat d'utilitzar la informació textual disponible en ambdues col·leccions fusionant-les amb la informació gràfica de la pàgina i analitzant el seu impacte sobre els resultats.

Després d'experimentar amb diferents arquitectures de xarxes convolucionals, s'han trobat millores en els resultats base d'una de les tasques. Per una altra banda, la informació textual extreta ha ajudat a obtenir millores en els resultats d'ambdues tasques.

Paraules clau: Reconeixement de Formes; Processament d'Imatges; Documents manuscrits; Anàlisis de Maquetació ("Layout"); Xarxes Neuronals Artificials; Xarxes Convolucionals; Aprenentatge Automàtic.

Resumen

Existen grandes colecciones de manuscritos, las cuales contienen información muy valiosa sobre aspectos cruciales de la historia de nuestra sociedad. Existe tal cantidad de documentos que de forma manual se tardarían años, o incluso siglos, en poder extraer toda la información, cuya mayoría es textual. Debido a esto, se trata de utilizar técnicas de maquetación y reconocimiento de texto manuscrito de las imágenes de forma automática a fin de poder comprender mejor, y de manera más eficiente, la información que nos proporcionan estas colecciones.

Este Trabajo de Fin de Máster se ha centrado en el desarrollo y evaluación de diferentes técnicas de aprendizaje profundo para realizar la maquetación de páginas con alto valor histórico.

Por lo que este trabajo gira en torno a dos tareas. La primera, la segmentación de zonas en un corpus del siglo *XIV* al siglo *XIX*. Dicho corpus está compuesto mayormente por tablas, habilitando un posterior análisis para permitir realizar consultas estructuradas. La segunda tarea trata de la separación de registros en una colección del siglo *XIV* al siglo *XV* dictados por el rey de Francia. Dicha separación ayudaría a la búsqueda de temas concretos de la época, así como posibles sentencias escritas en dichos registros.

Además, se ha utilizado la información textual disponible en ambas colecciones para fusionarla con la información gráfica de la página y analizar así su impacto sobre los resultados.

Tras experimentar con diferentes arquitecturas de redes convolucionales, se han mejorado los resultados base en una de las tareas. Por otro lado, la información textual extraída del contenido textual de los documentos ha ayudado a obtener mejoras en los resultados en ambas tareas.

Palabras clave: Reconocimiento de Formas; Procesado de Imágenes; Documentos manuscritos; Análisis de Maquetación ("Layout"); Redes Neuronales Artificiales; Redes Convolucionales; Aprendizaje Profundo.

Abstract

There are large collections of manuscripts which contain very valuable information on crucial aspects of the history of our society. There are such large quantity of documents that it would take years, of even centuries, to manually extract all the information, most of it is textual. Due to this, there are automatic image document layout analysis and handwritten text recognition techniques that helps to get a better understanding –and more efficiently –of the information provided by these collections.

This Master Thesis has focused on the development and evaluation of different deep learning techniques to make the layout of pages with high historical value.

This work revolves around two tasks. The first one, the zone segmentation in a corpus of the century *XIV* to the century *XIX*. This corpus is mostly composed by tables, enabling a subsequent analysis to allow structured queries. The second task deals with the separation of records in a collection of the century *XIV* to the century *XV* dictated by the King of France. This separation would help the search for specific issues of the time, as well as possible sentences written in those records.

In addition, we have tried to use the textual information available in both collections used for merge this information with the graphic information of the page and checking its impact on the results.

We have experimented with different architectures of convolutional networks and the base results have been improved in one of the tasks. It also shows how the extracted textual information helps to obtain improvements in the results in both tasks.

Key words: Pattern Recognition; Omage Processing; Handwritten Documents; Layout Analysis; Artificial Neural Networks; Convolutional Networks; Deep Learning.

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

El documento escrito se usa desde la antigüedad como medio para almacenar y transmitir información. Existen registros de lenguaje escrito en diferentes soportes, algunos con miles años de antigüedad. Con el paso del tiempo, se han venido acumulando cantidades astronómicas de documentos, la gran mayoría de ellos manuscritos, que contienen información valiosísima sobre la historia de la humanidad y sus desarrollos sociales, políticos y culturales, siendo estos los testigos más importantes en la herencia cultural común. Desde hace algunas décadas, bibliotecas y archivos de todo el mundo se esfuerzan en producir copias facsímiles de estos documentos en forma de imágenes digitales de alta resolución. La doble finalidad de ello es la de garantizar su preservación y facilitar su divulgación.

Sin embargo, la verdadera riqueza de estos documentos, es decir su contenido textual, sigue estando inaccesible para su uso primario: extracción de la información transmitida por el texto. Esta información se encuentra virtualmente enterrada bajo los billones de píxeles de las imágenes digitales obtenidas de estos documentos.

La forma más directa y obvia para facilitar el acceso, lectura, búsqueda y posterior conservación de la información es la transcripción. Ésta consiste en obtener el contenido de estos documentos en un formato de texto estructurado y de fácil manejo para un posterior proceso de dicha información y crear herramientas para realizar consultas sobre ésta.

La transcripción de estos documentos requiere, en primer lugar, de expertos altamente cualificados y que puede ser realizable a mano para colecciones de algunas decenas de imágenes. Esto acaba convirtiéndola en un proceso lento y muy caro. En el caso de conjuntos pequeños (digamos unos pocos cientos o miles) de imágenes, esta situación puede mejorarse hasta cierto punto mediante técnicas interactivas de Transcripción Asistida como las que se proponen en [Toselli et al. 2010]. Sin embargo para colecciones grandes (centenares de miles o millones de imágenes), cualquier forma de transcripción explícita, manual o semi-automática resulta completamente inasumible.

Actualmente existen técnicas de transcripción de imágenes de forma totalmente automática, las cuales han reportado resultados con gran precisión [Puigcerver 2018] (vemos con más detalle en una de estas técnicas en la sección 2.3). Para hacer un uso eficaz de estas técnicas es necesario una correcta interpretación y análisis de la maquetación (*document layout analysis* en inglés, DLA de aquí en adelante) de la imagen y, posteriormente, extraer las líneas de texto.

Existen textos que implican mayor dificultad en los cuales la precisión disminuye drásticamente. Esto puede ser debido a varias razones. Dichas razones pueden ser: un mal resultado del DLA, una mala extracción de líneas o una mala calidad de éstas debido al interlineado. Otro problema es el orden de lectura ya que éste está relacionado con la correcta interpretación de la maquetación del texto en la imagen.

Definimos el orden de lectura como el orden natural con el que un lector lee una página. El orden natural suele ser de arriba a abajo y de izquierda a derecha, empezando por el título y saltando a las notas marginales y de pie de página cuando sea necesario. El orden de lectura tiene en cuenta los saltos entre columnas, en el caso de que las tenga. También pueden existir tablas con un índice en la cabecera e información en las celdas de contenido.

Actualmente, los sistemas de transcripción automática consiguen un buen resultado cuando se consigue una perfecta maquetación, una buena detección de líneas y un buen orden de lectura. Esto se puede conseguir cuando las colecciones de datos son de tamaño reducidos pero se vuelve impracticable cuando las colecciones aumentan a cientos de miles, o incluso millones.

Las aproximaciones tradicionales para el análisis automático de maquetación están basadas en técnicas de procesado de imágenes Estas técnicas permiten detectar zonas de una imagen que constituyen bloques de texto. También es necesario separar los diferentes bloques de texto entre sí conforme su función en la maquetación, como pueden ser: títulos, cabeceras de sección, notas al margen, pies de página, numeración de páginas, texto entre líneas, entre otras. Esto se puede observar en la figura 1.1.

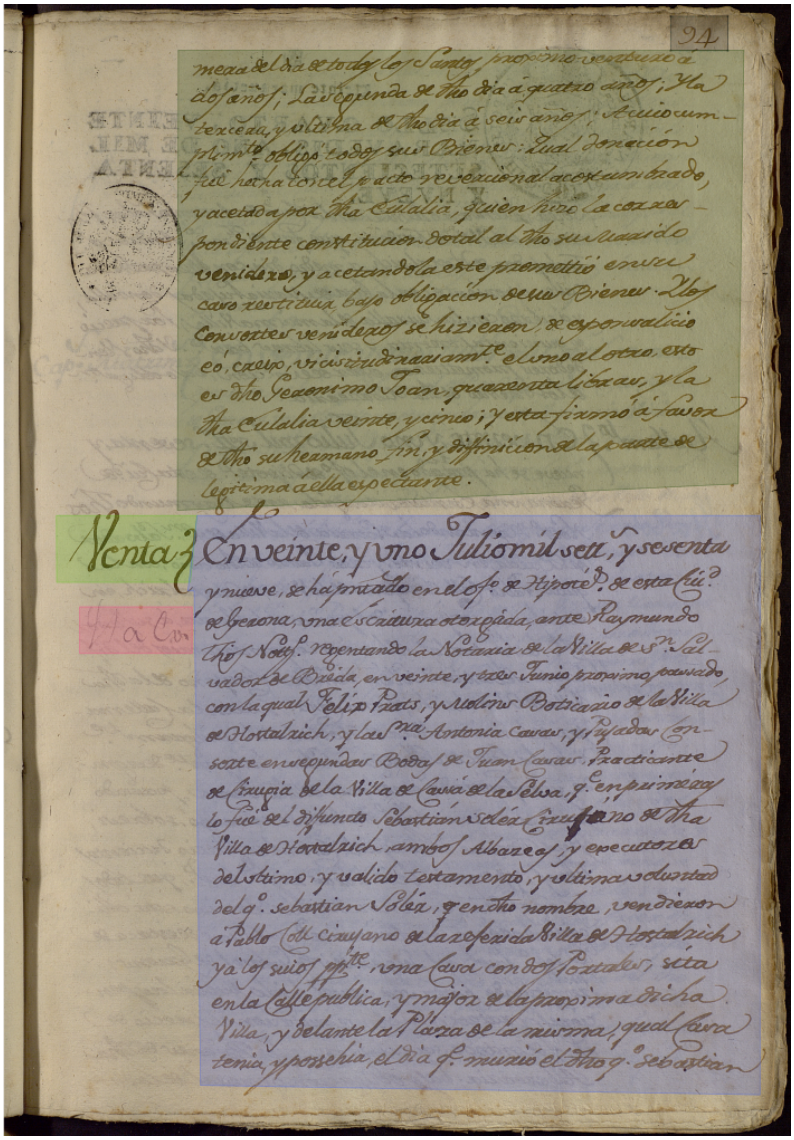


Figura 1.1: Ejemplo de maquetación. Se pueden diferenciar dos párrafos, en verde y azul, dos marginalias diferentes a la izquierda y, arriba a la derecha, el número de páginas.

Sin embargo, con gran frecuencia se encuentran imágenes de texto manuscrito en las cuales es imposible saber a qué bloque de texto pertenece una palabra, línea o incluso párrafo sin leer el texto en cuestión. Así pues, se presenta un interesante círculo vicioso según el cual si supiéramos analizar automáticamente la maquetación de una página con precisión, también podríamos obtener automáticamente transcripciones útiles. Sin embargo, de manera frecuentemente, tan solo teniendo esas transcripciones no es posible conseguir un DLA adecuado.

1.1 Estado de la cuestión

A lo largo del tiempo, se han usado muchos métodos para el DLA de documentos antiguos. Algunos trabajos se han basado en perceptrones multicapa (*MLP*, de ahora en adelante), explicados con detalle en la sección 2.1.

En [Bukhari et al. 2012] extraen características a partir de componentes conexas con una cierta ventana para determinar el contexto de cada componente. Utilizan un *MLP* seguido de una votación final para la clasificación de regiones (más específicamente, marginalias y cuerpo principal del texto) con algoritmos genéticos para determinar los parámetros del *MLP*.

En [Baechler and Ingold 2011] también utilizan componentes conexas para crear un sistema de tres niveles basados en clasificadores *MLP*. Cada nivel trata de extraer una segmentación de la imagen y, a su vez, refinarlo en el siguiente nivel en base a la misma imagen y el resultado anterior pero con mayor resolución.

En [Wei et al. 2013] comparan los *MLP* con máquinas de vector soporte (*SVM*) y modelos de mixturas de gausianas (*GMM)*. Para ello generan manualmente un vector de 87 dimensiones para cada píxel a través de sus vecinos, colores y otras características. Posteriormente comparan los clasificadores. Los mejores resultados, dependiendo del corpus, fueron los de *SVM* y *MLP*. Finalmente, pese a unir los clasificadores mediante una votación, los resultados no se consiguieron mejorar.

Por otro lado, también se han utilizado *Conditional Random Fields* (*CRF*) usando características relativas a la posición de cada pixel [Fern and Terrades 2012]. Dichas características se sacan de las relaciones entre clases de dichos píxeles. Por ejemplo, si después de una cabecera siempre va una celda, se añade como característica. Añadiendo estas caracteristicas se obtienen mejores resultados.

Recientemente, el estado de la cuestión se ha centrado más en redes completamente convolucionales (explicadas con más detalle en la sección 2.1) en las cuales se realizan varias tareas de forma simultanea: segmentación de líneas, extracción de imágenes, etc. Esto es asi debido al auge y la importancia que han adquirido las técnicas de aprendizaje profundo. *DhSegment* [Ares-Oliveira et al. 2018] es una herramienta que utiliza estas técnicas mediante una red neuronal completamente convolucional. Dicha herramienta realiza dos pasos: primero extrae la probabilidad de pertenecer a cada clase para cada píxel y después, transforma dichas probabilidades a la salida deseada, dependiendo de la tarea. Una vez entrenada, esta herramienta permite extraer de la página: las líneas, las zonas de texto, las decoraciones y las fotos.

En [Grüning et al. 2018] se presenta otra herramienta para realizar DLA de documentos antiguos mediante la extracción de líneas, la cual está dividida en dos fases. La primera fase se centra en entrenar una red neuronal completamente convolucional basada en una *U-Net* (arquitectura explicada en la sección 3.4 con detalle) llamada *ARU-Net*. Dicha red se basa en añadirle bloques de atención en la parte del *decoder* de la *U-Net* y bloques residuales en cada capa. En la segunda fase, se realiza una clasificación calculando superpíxeles y extrayendo así las líneas finales.

Otra herramienta es *P2PaLA* [Quirós 2018], la cual a su vez, está dividida en dos fases. La primera, está formada por una red neuronal completamente convolucional con el objetivo de extraer las probabilidades de cada píxel. Y, la segunda se centra en realizar un post-proceso buscando las zonas y las líneas a extraer. Dicha herramienta se explica con detalle en la sección 2.1 dado que parte de este trabajo se basa en la red neuronal de ésta.

1.2 Motivación

La utilidad del DLA de los documentos antiguos manuscritos no solo radica en la propia transcripción, sino también en la consulta de forma rápida y eficiente en un conjunto de éstos, así como determinar el correcto orden de lectura. Por ejemplo, gracias a ello, se podría acudir a un número de página concreto de un capitulo determinado en un libro antiguo, extraer todos los títulos de las diferentes páginas, buscar el año de nacimiento de una persona en una tabla o contar el número de *actos* de una obra. En este sentido, los actos en la colección de *Chancery*, utilizada en este trabajo, son registros dictados por el rey de Francia en el siglo XIV. De aquí en adelante nos referiremos a dichos registros como actos.

En este trabajo se propone resolver la maquetación en dos tareas diferentes: la detección de zonas en tablas y la separación de actos.

La detección de zonas en tablas es útil y necesaria como tarea previa a la consulta de dicha información, tal y como se hace en [Lang et al. 2018]. No obstante, en [Lang et al. 2018] dichas consultas se realizan con restricciones únicamente geométricas. Sin embargo, conviene destacar que en el caso de la detección y segmentación de tablas, cabría añadir otro tipo de restricciones para así mejorar dichas consultas. A modo de ejemplo, si se pretende consultar el número exacto de personas que se casaron en el año 1895, se debería buscar en un primer lugar dentro de la cabecera, la columna del año de cada boda y, posteriormente, buscar en el contenido de dicha columna el año mediante la aplicación de un filtro con la información buscada.

1.3 Objetivos

Uno de los dos objetivos principales de este trabajo es la exploración de diversas alternativas de arquitecturas de redes neuronales artificiales (ANN, por sus siglas en inglés) basadas en capas convolucionales. El otro objetivo principal es analizar el impacto del uso de índices probabilísticos sobre las hipótesis de palabras que pueden estar escritas en cada píxel de cada imagen. Estos índices probabilísticos podrían ser de gran importancia para desambiguar bloques de texto diferentes pero que resultan indistinguibles por su aspecto visual.

Para complementar estos objetivos es necesario marcar una serie de metas:

La primera consiste en obtener un resultado base en las dos tareas propuestas, es decir, en la segmentación de tablas y en la separación de actos.

La segunda consiste en explorar arquitecturas alternativas de redes neuronales convolucionales para tratar de mejorar el modelo base *P2PaLA*.

La tercera consiste en analizar el impacto del uso de índices probabilísticos.

Y por último, la cuarta meta consiste en explorar las diferentes arquitecturas de redes neuronales convolucionales junto con los índices probabilísticos y ver, a continuación, como afectan éstos a los resultados dependiendo de la arquitectura utilizada.

CAPÍTULO 2
Fundamentos

En este capítulo se explican algunas técnicas y operaciones básicas del campo que abarca este trabajo, justificando su uso y origen.

2.1 Redes neuronales artificiales

Las redes neuronales artificiales son una rama popular del aprendizaje automático. En este trabajo, se utiliza de manera amplia un caso especial de redes neuronales. Dichas redes neuronales son las llamadas redes neuronales convolucionales (*Convolutional Neural Networks* o *CNN* de aquí en adelante). Por lo que a continuación se explican qué se consideran redes neuronales artificiales de manera general para, posteriormente, analizar en que consisten las *CNN*.

2.1.1. Orígenes

El cerebro humano contiene una cantidad significativa de neuronas. Dichas neuronas están conectadas entre si de diferente forma, creando así una compleja red de señales de transmisión de información. Las ANN fueron propuestas para mimetizar las funciones neuronales del cerebro humano. En el año 1957 se propuso la unidad básica por la cual se constituye la base de las redes neuronales más clásicas: el perceptrón [Roseblatt et al. 1957].

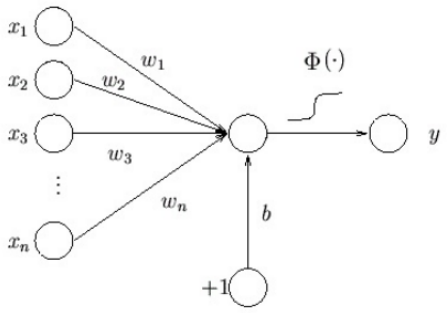


Figura 2.1: Representación gráfica de un perceptrón.

El perceptron tiene una serie de pesos ponderados $w_{1..n}$ y los combina con una serie de entradas que recibe $x_{1..n}$. Si la combinación excede cierto umbral,

devuelve una salida u otra. Para dicha salida se utiliza una función de activación $\Phi(\cdot)$, la cual suele devolver un valor en un rango de $]0,1[$ o $]-1,1[$. Dado que la derivada de la función de activación $\Phi(\cdot)$ se usará posteriormente para entrenar la red neuronal, es conveniente que dicha derivada se pueda expresar en términos de la función original. En la figura 2.1 se puede observar de forma gráfica la representación del perceptrón como neurona y en la ecuación 2.1 de forma matemática.

$$y = \Phi(s) = \Phi(\sum_{i=1}^{n} w_i x_i + b) \tag{2.1}$$

Las redes neuronales combinan múltiples perceptrones, típicamente en diferentes capas. Dichas redes son conocidas como perceptrones multicapa (figura 2.2), la versión más básica de las redes neuronales. Los preceptrones multicapa consisten en diferentes perceptrones repartidos en diferentes capas donde, los perceptrones de la capa l están conectados a todos los perceptrones de la capa $l+1$. Así sucesivamente hasta llegar al final de la red.

Estas redes tienen tres tipos de capas, las cuales están bien diferenciadas en la figura 2.2: la capa de entrada (rojo), la cual recibe los datos sin modificar. A continuación, están las capas ocultas [1] (en verde), en las cuales se genera el mayor cómputo de la red. Finalmente, se llega a la capa de salida (en azul), la cual clasifica en las diferentes clases mediante una activación típicamente diferente a la de las capas ocultas. Un perceptrón multicapa (*Multilayer perceptron* o *MLP* de aquí en adelante) con al menos una capa oculta puede funcionar como un aproximador universal de funciones, es decir que puede construir cualquier función [Hornik 1991].

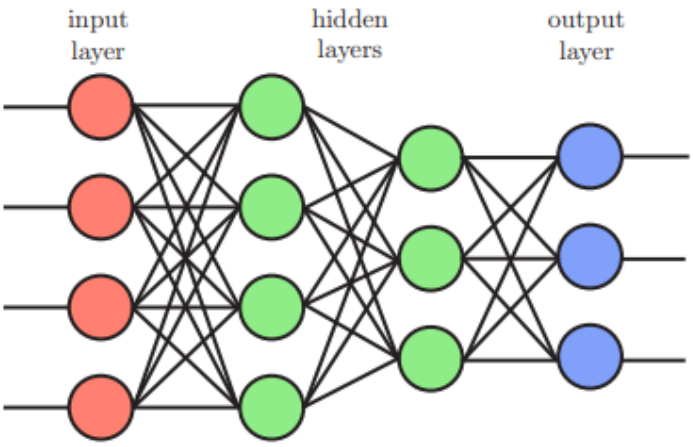


Figura 2.2: Representación gráfica de un perceptrón multicapa totalmente conectado.

2.1.2. Funciones de activación

La función de activación $\Phi(\cdot)$ determina como será la salida final de una neurona. Es importante seleccionar una función de activación adecuada para cada problema. Aunque la elección de cada función en cada neurona es una

[1] Se les llama ocultas dado que su activación no es 'vista' directamente por el entorno (entrada o salida))

área de investigación muy activa, todavía no existe una guía definitiva de unos principios teóricos de como elegir cada una de éstas. A continuación se muestran algunas funciones de activaciones comúnmente utilizadas.

Una de las funciones más utilizadas es la *Rectified Linear Unit* o *ReLU* (ecuación 2.2). Dicha activación es una opción muy común dado que, de forma empírica, se ha demostrado su utilidad en multitud de trabajos.

$$g(z) = \max(0, z) \tag{2.2}$$

Una variante de la función de activación *ReLU* es la `Leaky ReLU`, definida en la ecuación 2.3. Dicha función es una generalización basada en un salto α cuando $z < 0$. Este salto permite que la función tenga valores menores a 0, haciéndola más flexible.

$$g(z) = \max(0, z) + \alpha \min(0, z) \tag{2.3}$$

Otras funciones de activación son la sigmoide, ecuación 2.4, y la tangente hiperbólica (*tanh*), ecuación 2.5. Estas dos funciones están relacionadas entre si como se señala en la ecuación 2.5.

$$g(z) = \sigma(z) = \frac{1}{1 + e^{-x}} = \frac{e^x}{e^x + 1} \tag{2.4}$$

$$g(z) = \tanh(z) = \frac{\sinh(z)}{\cosh(z)} = \frac{(e^x - e^{-x})/2}{(e^x + e^{-x})/2} = 2\sigma(2z) - 1 \tag{2.5}$$

Algunas de las funciones de activación no son derivables en todos los puntos (por ejemplo, la función *ReLU* no es derivable en $z = 0$). Aunque pueda parecer que esto inhabilita la función para usarla con algoritmos de descenso por gradiente (explicados en la subsección **??**), en la práctica estos algoritmos funcionan bien [Goodfellow et al. 2016, p. 186].

2.1.3. Técnicas para regularizar

La red neuronal trata de buscar patrones y aprender como clasificar los datos. A veces, esta red neuronal puede llegar a memorizar los datos de entrenamiento (sobre-entrenamiento). Pese a obtener muy buenos resultados en estos datos de entrenamiento, supone un problema con los nuevos datos que deba procesar. Los modelos adquieren capacidad de memorizar los datos sin llegar a generalizar debido a que tienen demasiados parámetros a entrenar.

Para prevenir que una red neuronal no sobre-entrene y se aprenda los datos de forma sistemática se utilizan técnicas de regularización. Las dos más comunes y utilizadas en este trabajo son el *dropout* [Srivastava et al. 2014] y *batch normalization* [Ioffe and Szegedy 2015]. También se utiliza *data augmentation* para, entre otras funciones que explicamos en este capítulo, evitar este sobre-entrenamiento.

Dropout es una técnica computacionalmente más económica y muy eficaz a la hora de regularizar. Dicha técnica permite anular (poner a 0), a la hora de entrenar, algunas conexiones de cada neurona de la red con cierta probabilidad. Esto crea un efecto similar a una técnica de *bagging* (tener diferentes modelos para aprender una mejor predicción evitando el sobre-entrenamiento), donde

se incluyen diferentes entrenamientos de distintos modelos para compararlos posteriormente. Esto se ilustra en la figura 2.3 en la cual, en el caso de una ANN pequeña, al desactivar diferentes conexiones entre las neuronas, conseguimos varias combinaciones. Mediante este mecanismo, se podría probar las diferentes sub-redes mostradas en la imagen en cada iteración.

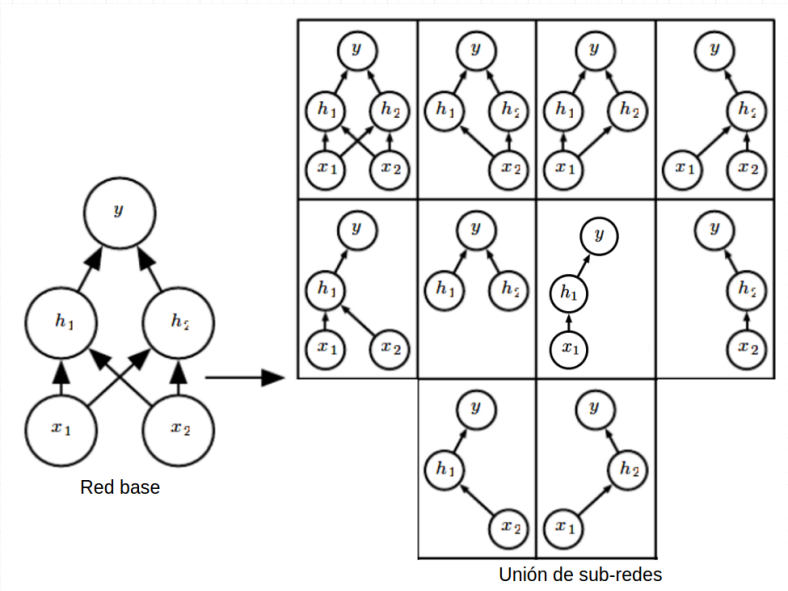


Figura 2.3: Efecto del *Dropout*. Figura basada en figura del capítulo 7 del libro *Deep Learning*, [Goodfellow et al. 2016, p. 252]. Para una red muy simple se pueden extraer, mínimo, 10 posibles combinaciones, creando 10 sub-redes.

Otra técnica ampliamente utilizada es *batch normalization*. Dicha técnica se aplica a cada mini-batch, es decir, sobre cada pequeño conjunto de datos en los que dividimos el conjunto de entrenamiento y la red neuronal procesa dichos datos. Esta técnica reduce la variación de la co-varianza entre las muestras y, al mismo tiempo, evita que los valores de la post-activación ni sean demasiado altos ni demasiado bajos.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Figura 2.4: Transformaciones de *Batch normalization*. Figura extraída del artículo original, [Ioffe and Szegedy 2015]. Transformación de *batch normalization* aplicado a la activación x sobre un mini-batch

En la figura 2.4 se puede observar el pseudo-algoritmo de esta técnica sobre un *mini-batch*. Esta técnica aumenta ligeramente el número de parámetros de la red en cuatro por cada capa, dos de ellos entrenables (γ, β) y dos de ellos no entrenables (media y varianza). La salida normalizada es multiplicada por γ (desviación estándar) y le es sumada β (la media).

Por otro lado, también tenemos *data augmentation*. Esto es un conjunto de técnicas que son utilizadas para aumentar el número de muestras diferentes del corpus de entrenamiento a partir de modificaciones sobre las propias imágenes existentes. Esto permite a los modelos no memorizar el conjunto de datos de entrenamiento y, por ende, no sobre-entrenar consiguiendo así una mejor generalización.

Varias técnicas son utilizadas para realizar *data augmentation* sobre imágenes. De dichas técnicas, las utilizadas en este trabajo son las siguientes:

- Translación: mover la imagen con un cierto porcentaje hacia un eje en concreto.
- Rotación: girar la imagen hasta un cierto número de grados sin pasar de 90º.
- *Shear*: mover la imagen para verla desde otro ángulo.
- Deformaciones elásticas: estirar la imagen hacia ciertos ángulos calculados [Simard et al. 2003] para así deformar dicha imagen.

Dichas técnicas mejoran notablemente el rendimiento de la red, ayudando a generalizar mejor. Éstas se aplican de forma aleatoria sobre los datos con una probabilidad del 50 % en cada uno.

Otra técnica utilizada para prevenir el sobre-entrenamiento es *early stop* teniendo en cuenta la pérdida en un conjunto de validación. Dicha técnica detiene el aprendizaje de la red cuando la pérdida en dicho conjunto de validación no disminuye en cierto número de iteraciones.

2.1.4. Entrenamiento

Conviene recordar que una ANN está compuesta por un conjunto de parámetros, inicialmente aleatorios, los cuales se optimizan mediante el algoritmo de descenso por gradiente y una función de pérdida especificada. Esta función es un método para especificar como aprende los datos la ANN. Si las predicciones se alejan mucho de los resultados reales, de la función de pérdida obtendremos un valor alto. Dicho valor indica que los resultados no han sido buenos. Por consiguiente, la red tratará de disminuir este valor lo más cercano a cero posible.

Para disminuir los resultados de la función de pérdida se utilizan técnicas de optimización. El algoritmo comúnmente utilizado para optimizar funciones en aprendizaje automático es el descenso por gradiente (*gradient descent*) a partir de ahora). Este algoritmo trata de actualizar los parámetros entrenables de nuestra ANN (o cualquier otro modelo) a partir de la derivada del error generado por la función de pérdida, ponderado por un factor de aprendizaje.

A fin de calcular el gradiente necesario para actualizar los parámetros (es decir, cuanto tenemos que actualizar cada parámetro según el resultado de nuestra función de pérdida y nuestra arquitectura de red) se utiliza el algoritmo de *retro-propagación* [Rumelhart et al. 1986] (más conocido como *back-propagation*).

Cuando un ANN acepta una entrada $\mathbf{x}$ y produce una salida $\hat{\mathbf{y}}$, la información fluye hacia adelante, pasando por todas las neuronas. A este fenómeno se le llama **propagación hacia adelante** (*forward propagation* en inglés). Cuando dicha propagación termina produce un coste escalar $J(\boldsymbol{\Theta})$ a partir de la función de pérdida definida. El algoritmo de *back-propagation* permite que la información, a partir del coste calculado, fluya hacia atrás en la red para calcular un gradiente.

El algoritmo de *back-propagation* nos permite calcular este gradiente. Mientras que, el algoritmo de *gradient descent* permite mejorar el aprendizaje de la red a partir de este gradiente. Es un error común confundir el algoritmo de *back-propagation* con todo el proceso de aprendizaje.

2.2 Visión por computador

El aprendizaje automático hace uso de la rama de visión por computador para poder extraer información significante a partir de imágenes o vídeos. En este trabajo se usan técnicas de segmentación de imágenes, un subproblema de visión por computador muy común.

2.2.1. Segmentación de imágenes

La segmentación de imágenes trata de dividir la imagen en regiones o categorías, donde cada píxel es asignado a una sola categoría, sin posibles solapamientos entre ellos.

Los problemas presentados en este trabajo se ven como un problema de segmentación de la imagen, donde a cada píxel se le asigna una probabilidad de pertenecer a cada clase (existiendo una clase por categoría) y, posteriormente, escoger la clase con mayor probabilidad.

Algunas técnicas clásicas de segmentación aplicadas al *document layout analysis* se pueden encontrar en [G 2000, Eskenazi et al. 2017] a modo de resumen mostramos las siguientes:

- *Clustering* basado en características básicas como la geometría, las sombras o elementos más básicos.
- Algoritmos basados en análisis de funciones, mayormente tratando de optimizar una función lo más posible a un objetivo, como pueden ser contornos activos o minimización de energía.
- Algoritmos de clasificación basándose en un conjunto de características, como pueden ser texturas. Para ello se necesitan datos etiquetados.

Algunas técnicas conocidas para clasificación utilizadas en segmentación son: *CRF*, *MLP*, *SMV* y redes neuronales convolucionales. En este trabajo entraremos en profundidad en estas últimas.

2.2.2. Redes neuronales convolucionales

El uso de redes neuronales clásicas, como las explicadas anteriormente en la sección 2.1.1, suele dar problemas en los problemas de visión por computador. Esto es debido a problemas de desvanecimiento del gradiente, es decir, que éste se vuelva extremadamente pequeño al propagarlo por la red. El número de parámetros aumenta dado que se debe tratar la imagen como un solo vector, cuando originalmente es una matriz. Esto puede causar sobre-entrenamiento y un rendimiento muy lento [Bishop 2006, p. 9] . Además, estos modelos no consiguen aprovechar toda la información espacial, como por ejemplo información de los vecinos de cada píxel. Esto dificulta la extracción de patrones o texturas por parte de la red al perder parte del contexto.

Las redes neuronales convolucionales, conocidas como *CNN*, son un tipo de redes especializadas en el tratamiento de datos en una rejilla, tanto en 1-D , tiempos en serie, como en 2-D , como puede ser un una rejilla de píxeles. El nombre de convolución proviene de la operación matemática que utiliza, la convolución, la cual es un tipo de operación lineal. Están inspiradas en un concepto biológico llamado campo receptivo, los cuales actúan como detectores de cierto tipo de estímulos, como pueden ser los ejes o patrones. Podemos observar un ejemplo de convolución en la imagen 2.5 .

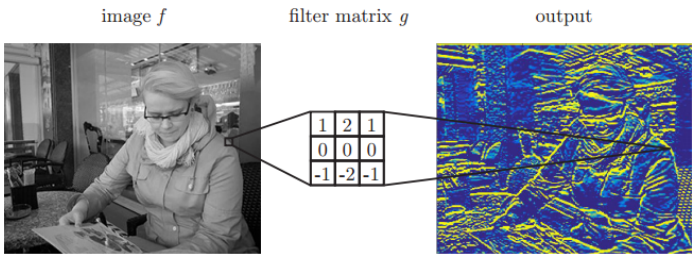


Figura 2.5: Convolución para extraer ejes horizontales.

Las operaciones convolucionales suelen usarse en más de un eje al mismo tiempo. En la ecuación 2.6 tenemos la definición de la operación como tal,

siendo X una imagen bidimensional como entrada y K un *kernel* bidimensional. Las convoluciones tienen tantos *kernels* como filtros de salida se desee, ya que aprende uno por cada uno de ellos, añadiendo más parámetros a la red.

$$S_{i,j} = (X * K)_{i,j} = \sum_m \sum_n X_{i-m,j-n} K_{m,n} \tag{2.6}$$

Una vez se juntan varios filtros convolucionales, combinados forman una *capa convolucional*, donde los *kernels* (que no dejan de ser matrices) son usados como parámetros de las neuronas y entrenados usando descenso del gradiente. Estos *kernels* se pasan por toda la imagen en forma de ventana y multiplicando los valores correspondientes de la imagen de entrada por los de éste, sumando al final y sacando un solo valor. Al pasar dicho *kernel* por toda la imagen se puede hacer dando pasos de un píxel o mas, siendo esto parametrizable en la función por una variable llamada *stride*. Si ponemos el *stride* igual a uno, se pasará por toda la imagen pero si aumentamos el valor de esta variable podemos hacer que la convolución de saltos sobre la imagen. Dependiendo del *stride*, del *padding* –otra variable que nos permite añadir píxeles extra a ambos lados de la imagen –y del tamaño del *kernel* calculamos el tamaño de la imagen de salida según las ecuaciones 2.7 y 2.8 para la imagen de salida en anchura como en altura, respectivamente. En dichas ecuaciones I^o es la imagen de salida, X es la imagen de entrada, X_w e X_h son la anchura y altura de la imagen de entrada, K es el *kernel* con K_w y K_h la altura y anchura del kernel, $\mathcal{P}$ se refiere al *padding* añadido y $\mathcal{S}$ al *stride*, con $1 \leq w \leq W$ y $1 \leq h \leq H$.

$$X_{w'} = \frac{X_w - K_w + 2\mathcal{P}}{\mathcal{S}_w} + 1 \tag{2.7}$$

$$X_{h'} = \frac{X_h - K_h + 2\mathcal{P}}{\mathcal{S}_h} + 1 \tag{2.8}$$

Por otro lado tenemos las operaciones de *pooling*. Dichas operaciones se utilizan para disminuir el tamaño de la imagen, tal y como haríamos aumentando el *stride* de la convolución, pero sin parámetros a añadir en la red. Estas operaciones tienen también una matriz y un *stride* la cual se va aplicando en forma de ventana sobre la imagen. Por ejemplo, una operación típica es el *max pooling*, el cual disminuye el tamaño de la imagen escogiendo el valor máximo en la ventana. Otra operación sería *average pooling*, la cual hace una media de los valores de la ventana. Por ejemplo, si aplicamos un *max pooling* con una matriz de 2×2 y un *stride* de 2×2 se obtiene una imagen con la mitad de tamaño. Importante mencionar que estas operaciones no afectan a los canales, por lo que se hace de forma individual para cada canal, sin interferir entre ellos.

No solo es necesario disminuir el tamaño de la imagen, sino que a veces se debe aumentar el tamaño de ésta. Para ello existen dos métodos. El primera es la operación inversa al *pooling*, la cual es llamada *upsampling*. Trata de aumentar el tamaño de la imagen dado una escala tratando de buscar, por ejemplo, un píxel cercano y duplicándolo o interpolándolo con otros. Se pierde calidad al hacer varias operaciones de este tipo.

Otro método aumentar el tamaño de la imagen es realizar convoluciones transpuestas, lo cual realiza una función similar a la de la convolución pero

la salida, en lugar de ser un solo píxel, es una matriz de píxeles, aumentando así el tamaño de la imagen. Con esto, se deja a la red aprender los pesos necesarios para realizar la operación, aunque añade un coste mayor en cuanto a la red. En las convoluciones transpuestas, a diferencia de las convoluciones explicadas anteriormente, la operación entre el *kernel* y la imagen se transpone, convirtiendo la ecuación 2.6 en la ecuación 2.9, y dependiendo del *stride* y del *padding* se aumenta el tamaño de la imagen X. La convolución transpuesta también es conocida como *convoluciones con stride fraccionado*.

$$S_{i,j} = (K * X)_{i,j} = \sum_m \sum_n K_{m,n} X_{i-m,j-n} \tag{2.9}$$

2.3 Índices probabilísticos

Keyword spotting consiste en buscar palabras en documentos o colecciones de documentos manuscritos con una confianza especificada, conocida como umbral. Para ello se utilizan los índices probabilísticos, los cuales indican, dada una imagen, una lista de palabras junto a sus coordenadas y la probabilidad de que ésta aparezca.

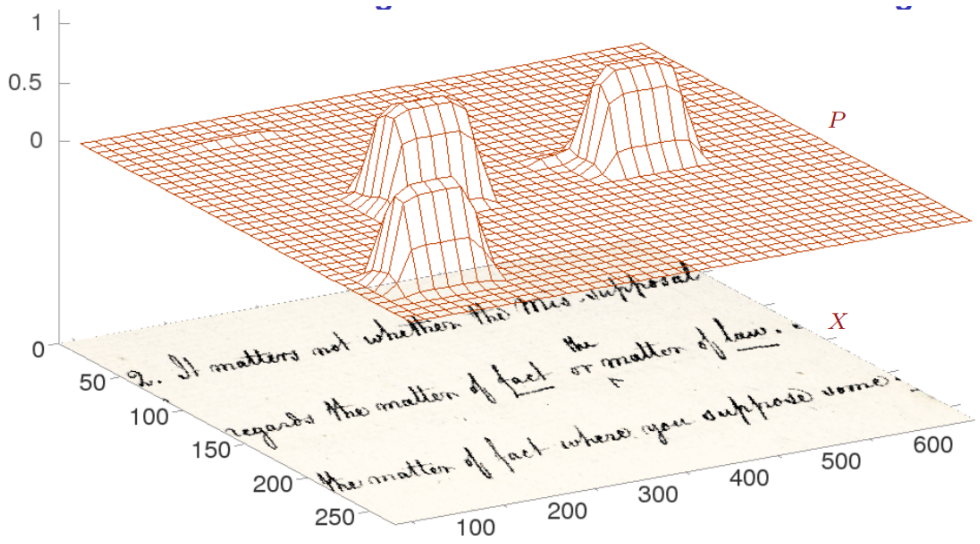


Figura 2.6: Índice probabilístico. Ejemplo de la probabilidad a posteriori para la palabra *matter*.

En la figura 2.6 podemos ver un ejemplo gráfico donde se ha calculado la probabilidad a posteriori $P(v|X,i,j)$, siendo $v =$ **"*matter*"** en una imagen X. Para calcular dichos índices se hace uso de redes convolucionales junto a redes recurrentes *LSTM* [Bluche et al. 2017].

2.3.1. Imágenes probabilísticas

Para poder interpretar los índices probabilísticos junto a la imagen en una red neuronal se han creado unas imágenes llamadas *imágenes de palabras* (o *class images*, *CI* de ahora en adelante).

A partir de la segmentación de zonas etiquetada manualmente para cada imagen se ha creado un clasificador, basado en la frecuencia de aparición cada

palabra en cada zona. De dicho clasificador se estima la probabilidad $P(c|v)$, donde dada una palabra v se obtiene la probabilidad de pertenecer a la clase c.

Al mismo tiempo, de los índices probabilísticos obtenemos el posteriorgrama de una imagen X [Vidal et al. 2019], el cual es la probabilidad de que la palabra v aparezca en el *bounding box* b que contiene el píxel (i, j), $b \in \mathcal{B}(i, j)$. En notación matemática como $P(v|X, i, j)$, siguiendo una distribución de probabilidad sobre todo el vocabulario V de la siguiente forma:

$$\sum_{v \in V} P(v|X, i, j) = 1, 1 \leq i \leq W, 1 \leq j \leq H, v \in V \tag{2.10}$$

Una forma de calcular $P(v|X, i, j)$ es considerando que cada palabra v está escrita en todos los posibles *bounding boxes* b de X, siguiendo la ecuación siguiente:

$$P(v|X, i, j) = \sum_{b \in \mathcal{B}(i,j)} P(v, b|X, i, j) = \sum_{b \in \mathcal{B}(i,j)} P(b|X, i, j) P(v|X, b, i, j) \tag{2.11}$$

donde $\mathcal{B}(i, j)$ es el conjunto de todos los *bounding boxes* de la imagen que contengan el píxel (i, j). En la figura 2.7 se ilustra el proceso de marginalización. $P(v|X, b, i, j)$ es la probabilidad de que v sea la única palabra escrita en la caja b, la cual incluye el píxel (i, j). De esta forma, por la independencia condicional de (i, j) dado b, la ecuación 2.11 se simplifica a:

$$P(v|X, i, j) = \sum_{b \in \mathcal{B}(i,j)} P(b|X, i, j) P(v|X, b) \tag{2.12}$$

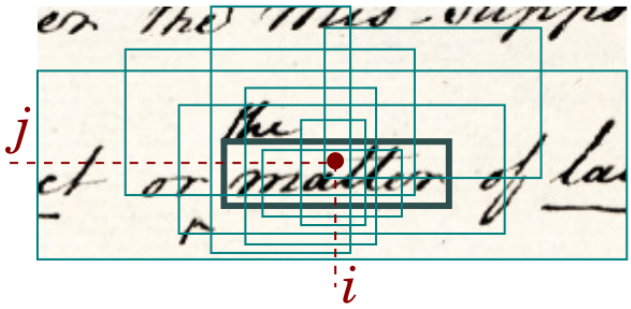


Figura 2.7: *Bounding boxes* $b \in \mathcal{B}(i, j)$. Para $v =$ "***matter***", la línea más gruesa indica mayor probabilidad del valor $P(v|X, b)$, mientras que en los otros *Bounding boxes* contribuyen con valores muy pequeños a la suma. . Figura extraída originalmente de [Vidal et al. 2019].

Las *CI* se explican con mayor detalle en la sección 3.1 , dado que forma parte del trabajo realizado..

2.4 Métricas

A continuación se explican las métricas usadas para evaluar los resultados de los diferentes problemas. No existe una métrica ideal para ambos problemas al mismo tiempo, dado que cada problema se evalúa de forma distinta.

2.4.1. Índice de Jaccard

Para evaluar la segmentación de zonas del corpus se utilizará la métrica llamada *Intersection over Union (IoU)*, conocida también como índice de Jaccard, el cual nos indica cuanto se solapa una predicción sobre el objetivo real. Una vez calculado se realiza una media sobre todas las imágenes del corpus.

De una forma simple, IoU calculará cuantos píxeles tienen en común una hipótesis y el objetivo como se muestra en la ecuación 2.13. En esta ecuación, I se refiere a IoU, t es el objetivo, datos ya etiquetados, y h es la hipótesis extraída.

$$I = \frac{|t \cap h|}{|t \cup h|} \tag{2.13}$$

La intersección $|t \cap h|$ nos indica cuantos píxeles hay en común entre las dos imágenes, mientras que la unión $|t \cup h|$ nos indica cuantos píxeles hemos clasificado como positivos entre las dos imágenes.

Esto es para una segmentación binaria, de dos clases. Cuando se trata de una segmentación en más de dos clases (más de una zona, en este caso) se usará la fórmula mostrada en 2.14, la cual es una aproximación de la fórmula 2.13, tal y como se hace en [Long et al. 2018], donde C es el número de clases diferentes, p_{ij} es el número de píxeles predichos de la clase i que pertenecen a la clase j y P_i es el número total de píxeles de la clase i.

$$I = \frac{\sum_i p_{ii} / (P_i + \sum_j p_{ji} - p_{ii})}{C} \tag{2.14}$$

2.4.2. *F-Measure*

Para la separación de zonas se utilizará la misma métrica que se suele usar en la detección de lineas. Se calculará la precisión (P) y el *recall* (R) para posteriormente calcular la media armónica (F1).

En [Gruning et al. 2018] se calcula el *recall*, R, de forma que indique que fracción de las línea de los datos reales (t) son detectados como líneas en la hipótesis con una cierta tolerancia, calculada de forma automática por el mismo algoritmo. En la precisión no se penalizan, a no ser que sea por un error muy grande, si la línea queda en su posición correcta o no. Se tiene más en cuenta que la línea exista a que esté bien situada.

Por otro lado, se calcula la precisión, P, en la cual si se penaliza más la posición de cada línea. Se calcula de forma automática un margen en el cual no se penalizará si no se sale de este y, a medida que la línea se aleje del margen, el valor de la precisión disminuirá.

Para el alineamiento de cada línea se utiliza un algoritmo *greedy* dado que normalmente no existe un orden de lectura disponible y no se puede aplicar un

algoritmo de programación dinámica. La solución encontrada será en la gran mayoría de casos la solución exacta.

Una vez se han calculado la precisión y el *recall* se procede a calcular la media armónica entre los dos valores, como se muestra en la ecuación 2.15.

$$F = \frac{2RP}{R+P} \tag{2.15}$$

CAPÍTULO 3
Trabajo realizado

En este capítulo se expondrá el trabajo realizado en la maquetación de imágenes de textos manuscritos antiguos. Para resolver las dos tareas enfocadas en este trabajo –la segmentación de tablas y la división de actos –se han realizado modificaciones en la herramienta *P2PaLA* [1]. Para ello, se han rediseñado los modelos de redes convolucionales, el post-proceso del resultado de dicha red y la forma de cargar los datos. Todo el código de este proyecto está disponible online [2].

3.1 Creación de imágenes con información textual

Para crear las imágenes con información textual, o CI (*class images*) de aquí en adelante, se sigue la ecuación 3.1, donde dado el píxel (i, j) se obtiene la probabilidad de que pertenezca a la clase c.

$$P(c|i,j) = \frac{P(c,x,y)}{P(i,j)} \tag{3.1}$$

Para resolver dicha ecuación se sigue a partir de la red bayesiana mostrada en la ecuación 3.2. Esta red indica que la probabilidad de que una palabra v pertenezca a la clase c, $P(c|v)$, depende de la propia palabra v. Dicha palabra está, a su vez, condicionada por su posición en la imagen, $P(v|i,j)$. Por ende, la probabilidad de la palabra v depende del píxel (i, j) en la imagen.

$$P(i,j,v,c) = P(i,j)P(v|i,j)P(c|v) \tag{3.2}$$

Si ahora desarrollamos la ecuación 3.1 llegamos a la ecuación 3.3. Finalmente, calculamos para cada píxel (i, j) de la imagen, la probabilidad $P(c|i,j)$, es decir, la probabilidad de que dicho píxel pertenezca a la clase que estamos buscando, c. El conjunto de palabras V, con $v \in V$, se define para cada tarea de forma distinta.

[1] https://github.com/lquirosd/P2PaLA
[2] https://github.com/JoseRPrietoF/P2PaLA.git

$$
\begin{aligned}
P(c|i,j) = \\
\frac{P(c,i,j)}{P(i,j)} = \\
\frac{1}{P(i,j)} \sum_v P(c,i,j,v) = \\
\frac{1}{P(i,j)} \sum_v P(i,j) P(v|i,j) P(c|v) = \\
\sum_v P(v|i,j) P(c|v)
\end{aligned} \tag{3.3}
$$

Dado al posible solapamiento entre los *bounding box* de los índices probabilísticos se realiza, posteriormente, una normalización teniendo en cuenta todos los *bounding box*.

En la sección 3.2 vemos algunos ejemplos de estas imagenes.

3.2 Segmentación de tablas

La primera tarea trata de una segmentación de zonas en un corpus con una gran cantidad de tablas. Dichas tablas no siempre tienen una forma homogénea. En ocasiones los bordes pueden estar bien delimitados y, en otras ocasiones, estos pueden no estarlo. Por lo que en este segundo caso cabría leer el contenido de dicha tabla para conseguir diferenciar las partes de ésta.

La tarea consiste, dada una imagen, en diferenciar lo que es la cabecera de una tabla, lo que es el contenido y, por descarte, lo que no es tabla (fondo o *background*). En la figura 3.1 vemos un ejemplo relativamente fácil de la tarea. En dicha imagen se puede distinguir de forma visual las distintas partes de la tabla y el fondo. En esta ocasión, la tabla está delimitada por líneas rectas mientras que en rojo podemos advertir la cabecera y en amarillo el contenido. Estas dos zonas son diferenciables entre si visualmente. Esto es así porque la cabecera usualmente suele estar siempre arriba y se suele delimitar con una doble linea o una sola línea de mayor amplitud, la cual se aprecia a simple vista.

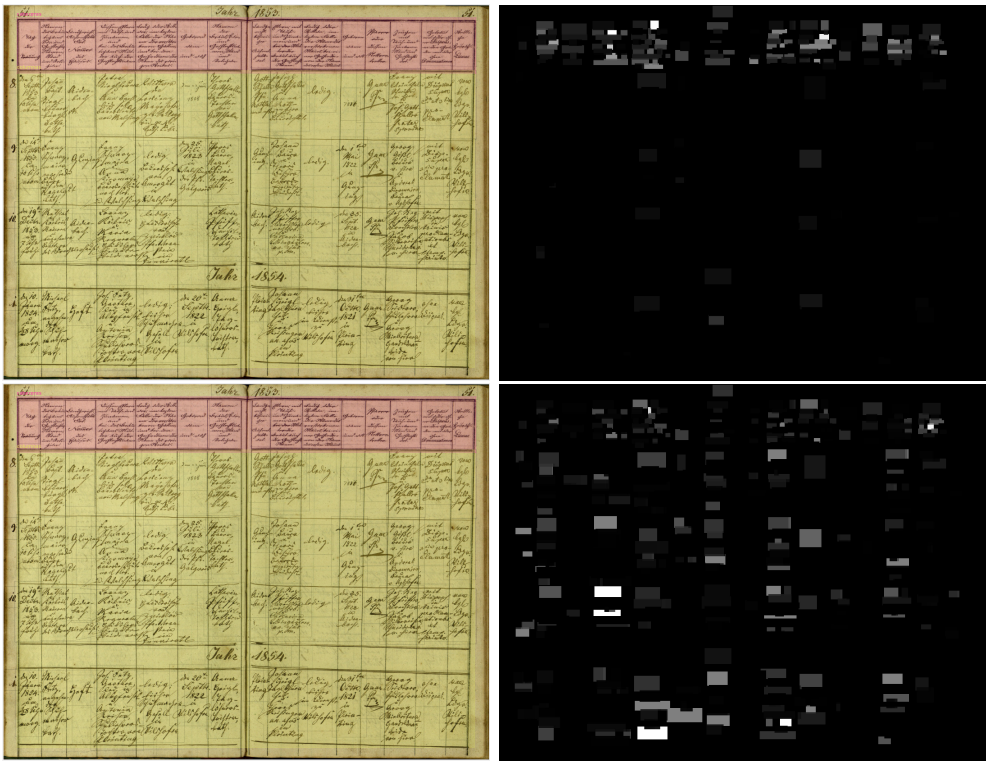


Figura 3.1: Ejemplo de tabla fácil de diferenciar. *Class images*.
A la izquierda vemos la imagen, repetida para una mejor visualización junto a la imagen de la derecha. A la derecha vemos las *class images*, arriba la de cabecera y abajo la de contenido.

En la figura 3.2 vemos una tabla similar a la precedente pero sin cabecera. La tabla forma la mayor parte de la imagen, es decir que hay muy poco contenido de fondo. Resulta difícil distinguir si existe una cabecera teniendo en cuenta solo el contenido visual, sin poder leer el contenido de la imagen.

En la siguiente tabla expuesta en la figura 3.3, contiene las tres clases posibles. No existe una línea clara de delimitación en la tabla, como tampoco entre la cabecera y el contenido de ésta. En rojo sombreado observamos la cabecera mientras que en amarillo el contenido. También existen imágenes donde no hay tablas, teniendo una sola clase, la de fondo.

Teniendo en cuenta que con solo el contenido gráfico de la imagen se hace difícil distinguir, sobre todo, las diferentes partes de la imagen, se ha tratado de utilizar el contenido semántico ya que de éstas se tienen los índices probabilísticos explicados en la sección 2.3.

Con esta información se ha tratado de utilizar técnicas multi-modal para fusionar la información útil de las dos fuentes, textual y visual. Para fusionar la información de dichas fuentes, se ha hecho uso de las *class images*, explicadas en un capítulo anterior. Éstas han sido creadas con la información semántica de cada página, a partir de las palabras más frecuentes de cabecera y contenido. Las palabras fuera del vocabulario se han ignorado.

Se ha tratado de sacar los mejores resultados posibles sin las *class images*, mejorando la arquitectura de la red neuronal. Posteriormente se han probado varias formas de fusión de la información con las *class images*. En la figura 3.1 vemos un ejemplo de *class images* extraidas respecto a la imagen de la misma figura en la parte superior.

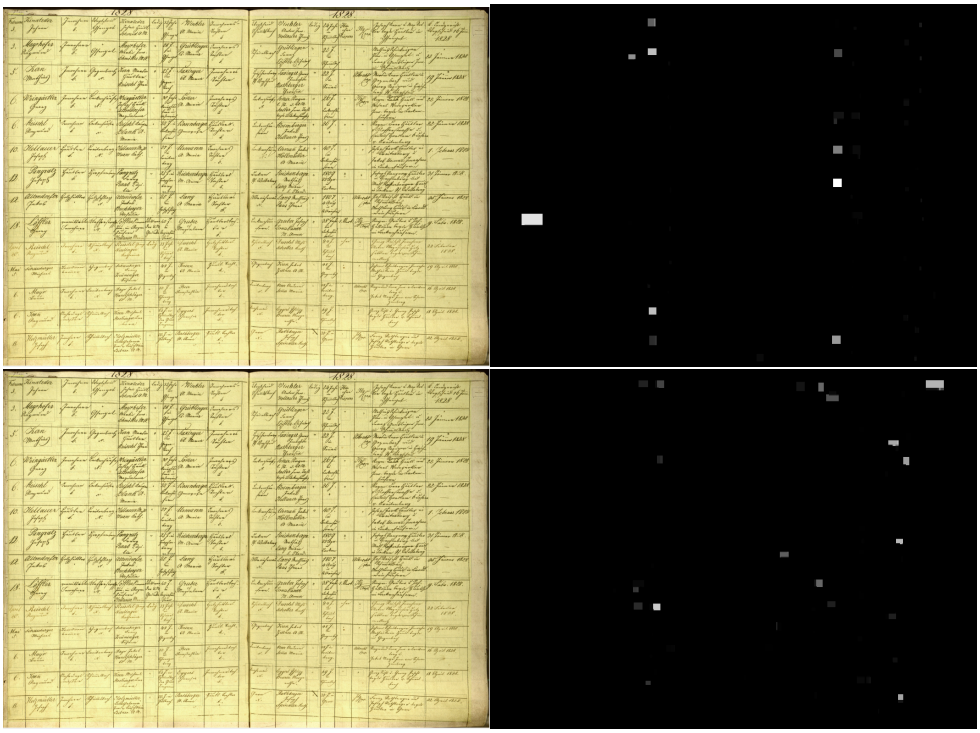


Figura 3.2: Tabla donde no hay cabecera. *Class images*. A la izquierda vemos la imagen, repetida para una mejor visualización junto a la imagen de la derecha. A la derecha vemos las *class images*, arriba la de cabecera y abajo la de contenido.

En la figura 3.2 vemos las *class images* extraídas de la imagen de la misma figura la cual está repetida en la izquierda. Se puede concluir que hay una tabla sin cabecera. En cuanto a la *class images*, en cuanto a la cabecera, no se puede distinguir claramente la presencia de una.

Como se puede observar, las *class images* parecen aportar información útil en algunos casos y en otros no tanto.

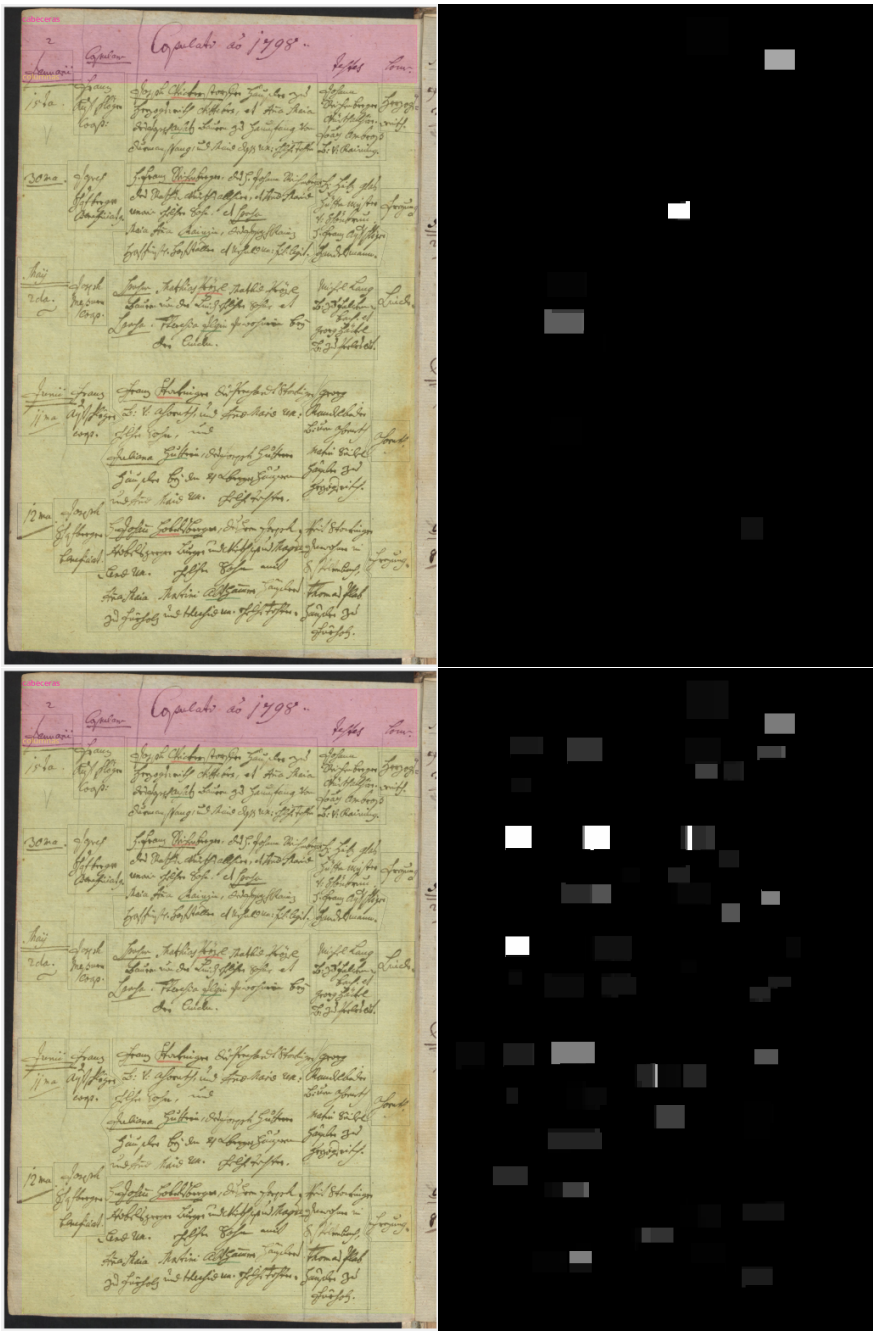


Figura 3.3: Tabla más difícil de diferenciar. *Class images*. A la izquierda vemos la imagen, repetida para una mejor visualización junto a la imagen de la derecha. A la derecha vemos las *class images*, arriba la de cabecera y abajo la de contenido.

3.3 División de actos

Esta segunda tarea se trata de separar, en cada página, los actos o registros por los cuales están formadas. Dichos actos o registros son del siglo XIV al siglo XV, explicados con mayor detalle en la sección 4.1.2. Para separarlos se crea una línea que ocupe todo el ancho de la imagen y se ha buscado el final del acto para dicha separación. En la figura 3.4 vemos una imagen con tres actos los cuales se pueden separar fácilmente de forma visual, dado que al final de cada acto hay una firma, un espaciado mayor y, además, al inicio de cada acto se muestra una marginalia a la izquierda.

En la figura 3.5 vemos otro ejemplo de separación de tres actos en la misma página. Ésta no tiene las mismas características que la imagen de la figura anterior, aunque si que se ve que al inicio de cada acto se empieza con una tipografía. Otro problema que dificulta el problema es el no poder diferenciar a simple vista si el último acto finaliza en esa página o continua en la siguiente.

Dado que de forma visual no parece un problema trivial en algunos casos se ha tratado de utilizar la información que proporcionan los índices probabilísticos para tratar de discriminar las fronteras entre actos no solo junto a la información que aporta la imagen sino también las palabras. Se han creado las *class images* con la información semántica de cada página, a partir de conocimiento experto de las palabras de inicio de acto y de final de acto, resultando dos imágenes, una para cada conjunto de palabras.

En la figura 3.4 vemos las *class images*, situados a la derecha de la imagen original. Como se observa, la imagen compuesta por las palabras de fin de acto (arriba a la izquierda), no nos aporta gran información, dado que las palabras no han acertado correctamente las finalizaciones de dichos actos. Al final de la imagen, por ejemplo, hay un fin de acto y la imagen probabilística no aporta información para éste. La imagen de palabras de inicio de acto pasa algo similar aunque si parece aportar algo más de información para el acto a media imagen.

En las imágenes de la derecha de la figura 3.5 vemos otro ejemplo. En esta imagen de probabilidades vemos como si se pueden distinguir algunos fin de acto (arriba), principalmente en el primer y segundo acto. En cambio la imagen de probabilidades de incio de acto (abajo) sigue sin aportar gran información.

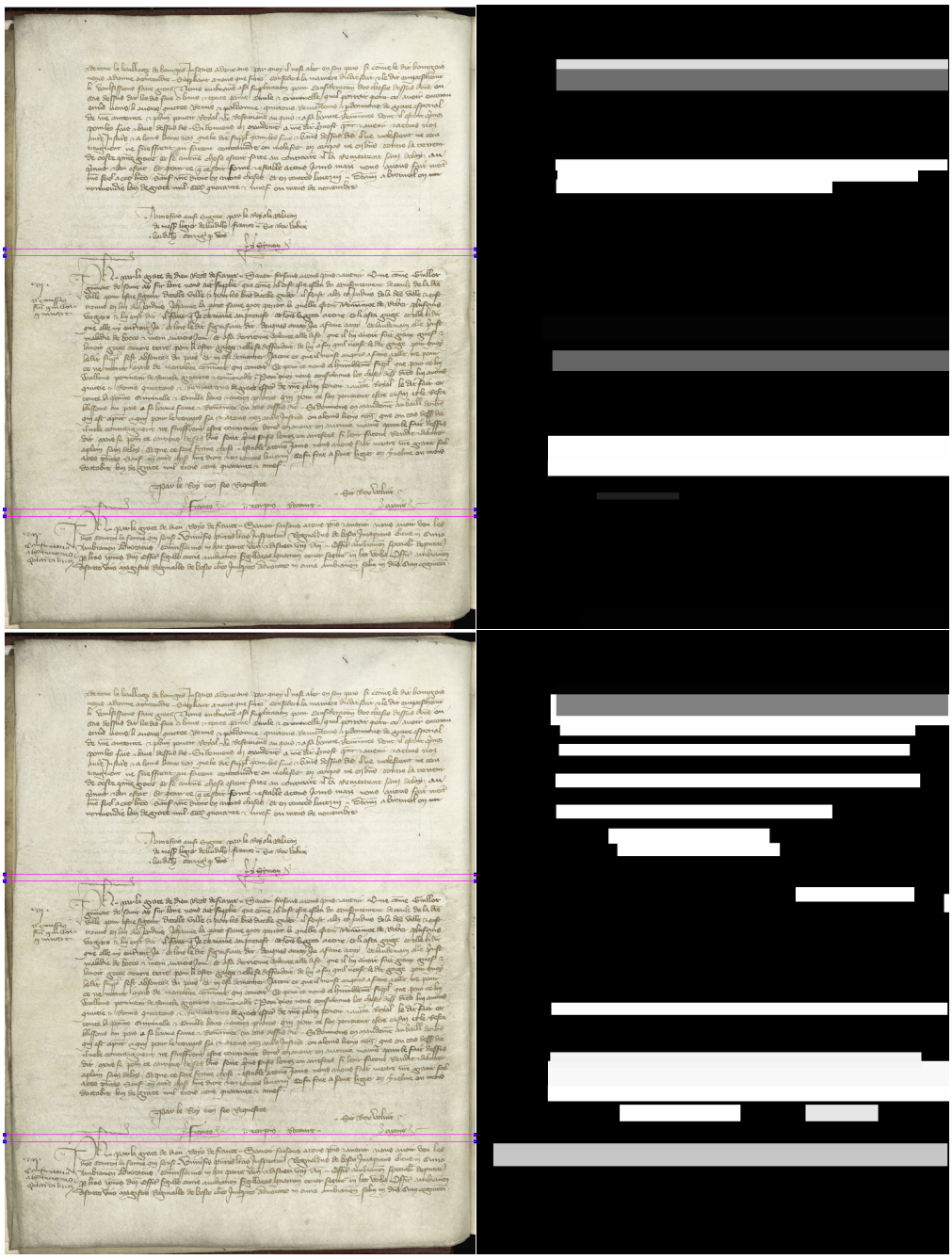


Figura 3.4: Ejemplo de separación de tres actos con *class images*. A la izquierda vemos un ejemeplo de fácil separación de tres actos. Arriba a la derecha podemos ver las palabras de final de acto mientras que abajo a la derecha vemos las palabras de inicio de acto. Se repite la imagen para una fácil visualización de las palabras de la izquierda sobre la imagen de la derecha.

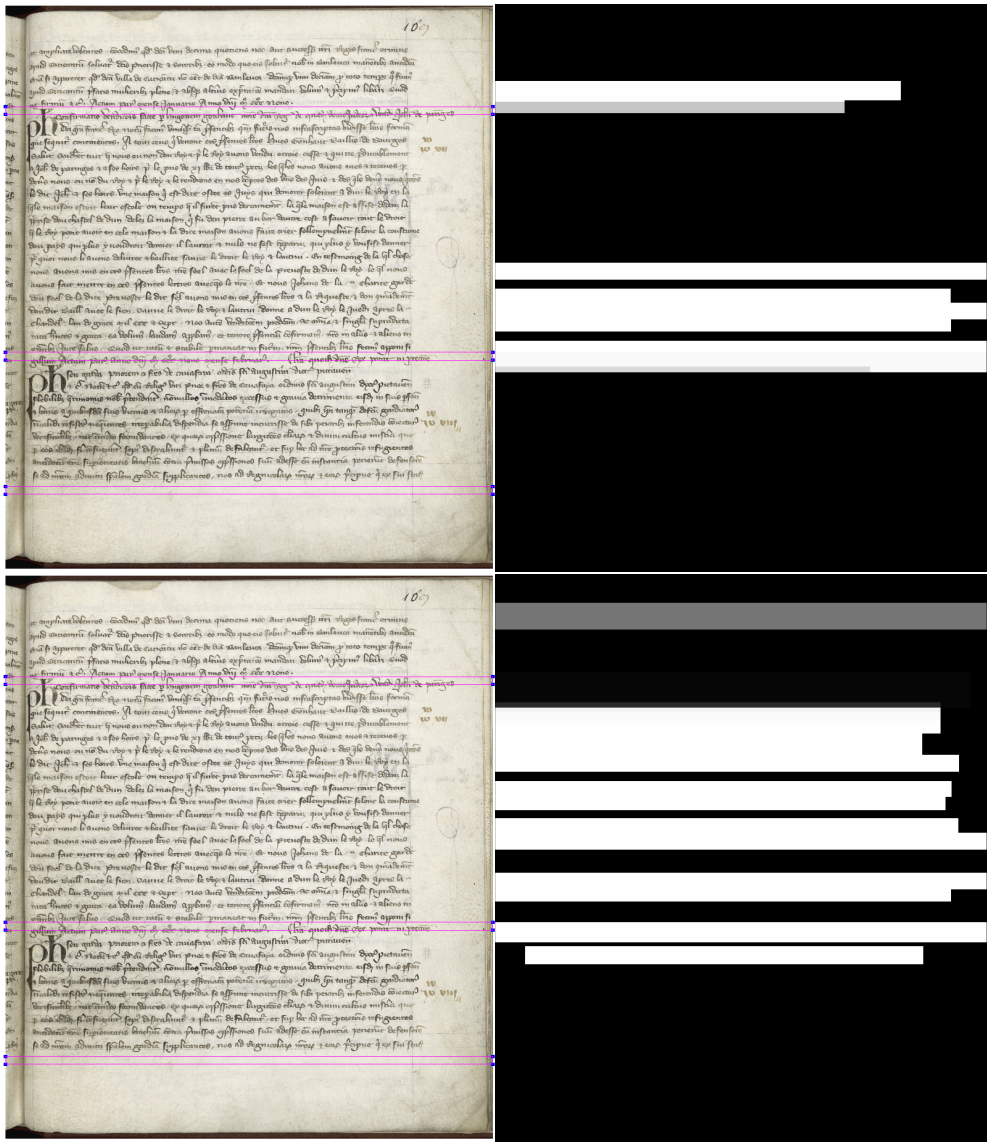


Figura 3.5: Ejemplo de separación de tres actos más complicado, con *class images*. A la izquierda vemos un ejemeplo de fácil separación de tres actos. Arriba a la derecha podemos ver las palabras de final de acto mientras que abajo a la derecha vemos las palabras de inicio de acto. Se repite la imagen para una fácil visualización de las palabras de la izquierda sobre la imagen de la derecha.

3.4 Arquitecturas implementadas

Ambos problemas se han abordado como una segmentación de regiones, tratando de clasificar todos los píxeles de la imagen en su correspondiente clase: las zonas de la tabla o los fin de acto en la segunda tarea. En este trabajo se ha re-implementado la red neuronal de [Quirós 2018], la cual sigue una arquitectura de *encoder* - *decoder* con bloques conectados a cada capa, llamados *Skip Connections*. dicha arquitectura es comúnmente llamada *U-Net* [Ronneberger et al. 2015].

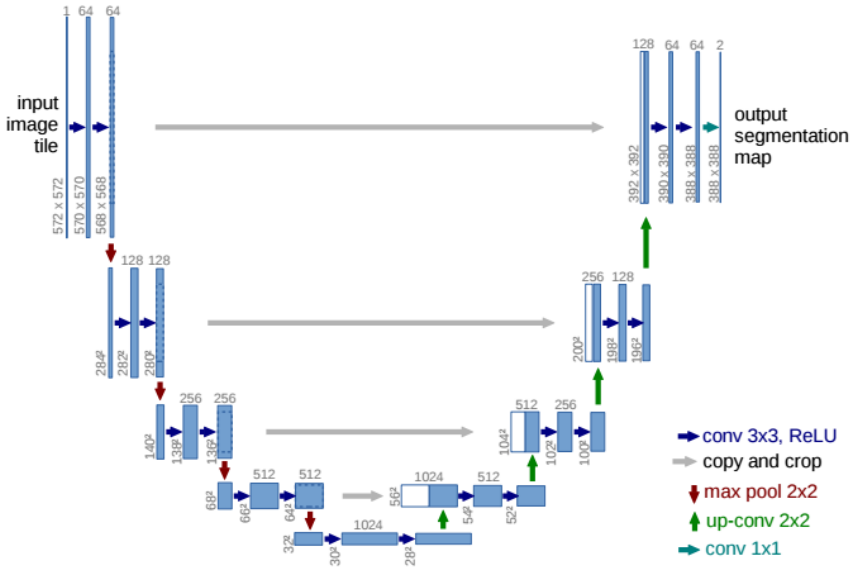


Figura 3.6: Arquitectura original de la primera *U-Net*. Imagen extraída de [Ronneberger et al. 2015].

En la figura 3.6 vemos la arquitectura de la primera U-Net que apareció, en el año 2015. En la parte izquierda vemos el *encoder*, parte de la red que trata de codificar la información contrayendo la imagen original a un tamaño más pequeño, aumentando el número de canales que tiene la imagen originalmente. Esta arquitectura consta de 5 niveles y en cada una de ellas se aplican dos convoluciones con un *kernel* de 3×3 sin *padding* y *stride* igual a 1, aplicando a cada una de ellas una función de activación *ReLU*. Después de las dos convoluciones se aplica una operación de *max pooling* de 2×2 con un *stride* de 2×2 para disminuir la imagen a la mitad, tanto en ancho como en alto. A cada capa se dobla el número de filtros de entrada de la anterior, empezando en la primera capa con 64 filtros (la cual recibe una imagen de 3 canales RGB) y acaba en la última capa con 1024 filtros y la imagen 4 veces más pequeña. La quinta y última capa hace de cuello de botella.

Seguidamente, empieza la fase de *decoder* a aumentar el tamaño de la imagen, realizando el mismo proceso que el *encoder* pero en lugar de *max pooling* usando *up sampling*, y finalizando en la última capa del *decoder* con una convolución de 1×1 y un número de filtros igual al número de clases deseada para la salida.

Una de las características principales de la U-Net son las *Skip Connections*, conexiones entre el *encoder* y *decoder*. En cada nivel el *decoder* recibe la salida del *encoder* de su mismo nivel y la concatena junto con la salida del nivel inferior del propio *decoder*. Es decir, la capa l_k del *decoder* recibe y concatena la salida de la capa l_{k-1} y la salida de la capa l_k del *encoder*. Con esto se consigue recuperar información de las niveles de mayor resolución del *encoder*, dado que al disminuir el tamaño de ésta se puede perder cierta parte de la información espacial, recuperando así parte del contexto.

3.4.1. Entrenamiento

El entrenamiento de la red tratará de, dado $\mathbf{x}$ como dato de entrada, estimar la distribución de probabilidad $p(\boldsymbol{y}|\boldsymbol{x})$, optimizando el problema de la ecuación 3.4.

$$\hat{\boldsymbol{y}} = \arg\max_{y} p(\boldsymbol{y}|\boldsymbol{x}) \tag{3.4}$$

Todas las arquitecturas probadas dan, como resultado en la última capa, dan una estimación de probabilidades de cada clase en base a la función *softmax* a nivel de píxel. Donde $p_c(i,j)$ se interpreta como $p_c(i,j) = p_c(y_{i,j} = 1|x_{i,j})$, esto es la probabilidad a posteriori que el píxel (i,j) de la imagen de entrada x pertenezca a la clase c. Se sigue la ecuación 3.5 donde $\Phi_c(i,j)$ indica la activación en el canal c en la posición del píxel i,j, $1 \leq i \leq W$, $1 \leq j \leq H$, siendo W y H la anchura y la altura de la imágen , respectivamente. C es el número de clases y $p_c(i,j)$ es la aproximación de la función máxima.

$$p_c(i,j) = \frac{\exp(\Phi_c(i,j))}{\sum_{c'=1}^{C} \exp(\Phi_{c'}(i,j))} \tag{3.5}$$

Es decir, $p_c(i,j) \approx 1$ cuando c tenga la máxima activación de $\Phi_c(i,j)$ y $p_c(i,j) \approx 0$ para el resto de c. Se aplica entropía cruzada como función de pérdida a dicha salida y penalizar la desviación de $p_{t(i,j)}(i,j)$ de 1.

La entropía cruzada penalizará, en cada píxel, la desviación de $p_{t(i,j)}$ de 1 usando la ecuación 3.6, donde $t(i,j)$ es una función que devuelve la clase correcta. Donde ω es el mapa de pesos ponderados para darle más importancia a algunas posiblemente des-balanceadas, siguiendo la ponderación por clase usada en [Quirós 2018], $\omega_c = \frac{1}{\log(\kappa+\pi_c)}$, donde κ es una constante y π_c es la probabilidad a priori de la clase c, con $c \in C$.

$$L(y,i,j) = \frac{-1}{WH} \sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{c=1}^{C} \omega_c y_{c,i,j} \log(p_c(i,j)) \tag{3.6}$$

Se han entrenado todos los modelos minimizando la entropía cruzada, usando *mini batch* con *SGD* y *Adam* [Diederik P. and Ba 2014] con un factor de aprendizaje de 0,001, $\beta_1 = 0{,}5$ y $\beta_2 = 0{,}999$. Se ha utilizado $\kappa = 1{,}02$ en la ponderació. Todas las pruebas se han entrenando hasta trescientas textitepochs utilizando una tarjeta gráfica *GeForce GTX 1080* con $8119MiB$ de memoria.

3.4.2. P2PaLA

Esta arquitectura se trata de una U-Net con algunas modificaciones en la estructura. Se pasa de 5 a 8 niveles, disminuyendo el tamaño de la imagen original hasta 7 veces. Esto es dado a que la U-Net original trabaja con imágenes de menor tamaño y, en tipo de problemas, es necesario el uso de imágenes de grandes resoluciones.

Cada capa en el *encoder* tiene un bloque compuesto por una convolución, seguido de *batch-norm*, *dropout* del 50 % y activación $LeakyReLU$. En el *decoder*

se sustituyen las activaciones *LeakyReLU* por *ReLU* y las convoluciones por convoluciones transpuestas, a fin de aumentar el tamaño de la imagen.

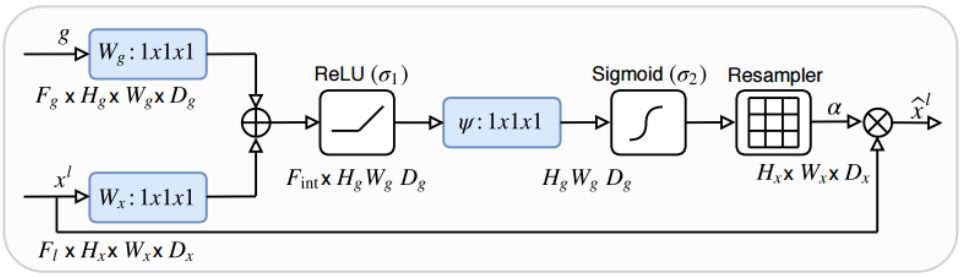
A diferencia del modelo U-Net original, en esta versión se utilizan en todas las convoluciones *kernels* de 4×4 con stride igual a 2 y padding igual a 1, teniendo una sola convolución en lugar de dos y eliminando tanto las operaciones *max pooling* como *up sampling*.

3.4.3. Squeeze attention

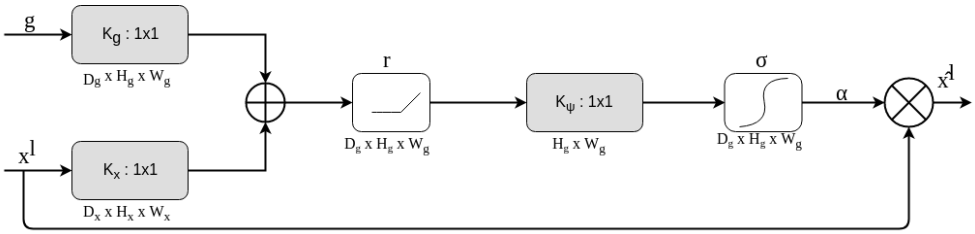
Los modelos de atenció fueron diseñados, orginalmente, para problemas de búsqueda de puntos de zonas en imágenes. Actualmente se ha extendido y demostrado su uso en diversos problemas, como en traducción automática u otras ramas de visión por computador.

Un modelo de atención trata de indicarnos en que parte de los datos de entrada hay más información relevante. La salida de dicho modelo son un conjunto de pesos del mismo tamaño que la imagen, en rango de cero a uno. Comunmente existen dos alternativas para usar estos pesos: sumarlos o multiplicarlos. Estos modelos se pueden añadir a cualquier capa de la red convolucional y se entrenan junto al entrenamiento de la red común, dado que formarà parte de ésta.

Se utiliza un modelo de atención llamado *Squeeze attention* [Oktay et al. 2018] en cada nivel del *decoder* del modelo de *P2PaLA*, modificando los pesos que se transmiten desde el *encoder* al *decoder* con las *Skip Connections*. Al reducir el tamaño de la imagen en el encoder y después volver a aumentarlo, en el decoder, se pueden perder ciertas características de la imagen como, por ejemplo, objetos pequeños. Con las *Squeeze attention* se trata de reducir estos errores y este modelo de atención trata de disminuir dichos errores.



(a)



(b)

Figura 3.7: *Squeeze attention*. Modelo de atención con las *Skip Connections* y la salida de cada capa del *decoder*. Imagen originalmente extraída de [Oktay et al. 2018].

En la figura 3.7a se puede observar el modelo de antención original. La notación de dicha figura no corresponde con la de este trabajo y, además, el modelo funciona sobre imágenes de tres dimensiones, mientras que aquí usamos imágenes de dos dimensiones. Por ello, hemos modificado el modelo a nuestra conveniencia, como se puede observar en la figura 3.7b.

Para calcular los pesos del modelo de atención, se calculan unos coeficientes, $\alpha_i \in [0, 1]$, llamados *Attention Gates* o AGs, tal y como se puede observar en la figura 3.7b al final. En la figura, r representa una función de activación $ReLU$, σ representa una función sigmoide, K_ψ una convolución con un kernel de tamaño 1×1, sin variar el tamaño del tensor en altura ni anchura pero disminuyendo los filtros a solo uno. Las dos entradas del modelo son g y x^l, de tamaño $(D_g \times H_g \times W_g)$ y $(D_x \times H_x \times W_x)$, respectivamente. K_g y K_x representan convoluciones con un kernel de tamaño 1×1 y con g y x filtros de salida, respectivamente.

La salida de los AGs es una multiplicación punto a punto en un mapa de características y coeficientes de atención $\hat{x}^l_d = x^l_d \cdot \alpha^l$, donde d es cada filtro, con $d \in D$ y l es la capa, con $l \in L$.

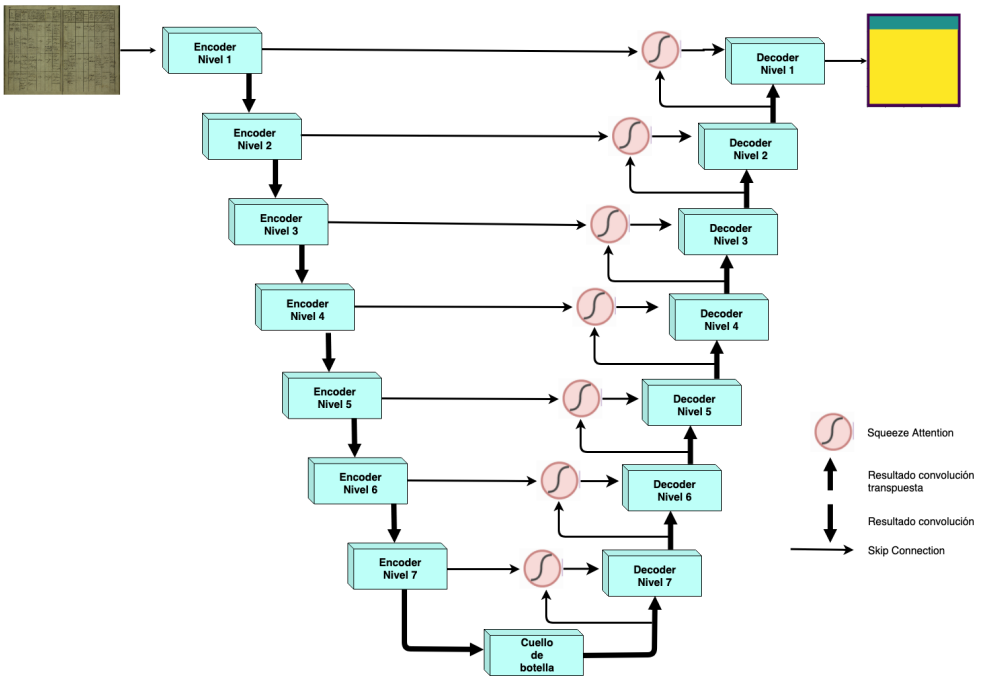


Figura 3.8: *U-Net con Squeeze attention* aplicado.

$$q^l = K_\psi * r(K_x * X^l + K_g * g^l) \tag{3.7}$$

$$\alpha^l = \sigma(q^l(x^l, g^l)) \tag{3.8}$$

Tal y como se muestra en la ecuación 3.7, con la función q^l obtenemos el resultado de las convoluciones de la figura 3.7b en la capa l, se aplican dos convoluciones de forma separada a las dos entradas: la entrada g, que es la imagen que proviene de la respectiva *Skip Connection*, y la entrada X, la imagen saliente del nivel inferior del *decoder*. Estas convoluciones son con kernel 1×1 y con un solo canal de salida, a fin de poder disminuir la dimensionalidad del tensor en profundidad, transformando todos sus filtros en uno solo. El

resultado de estas dos convoluciones se suma y se aplica una función de activación *ReLU*, σ_1, y se le aplica otra convolución de 1×1 con un canal de salida, ψ^T, con función de activación sigmoide, σ_2, mostrada en la ecuación 3.8. Estos pesos entre 0 y 1 ponderarán los pesos provenientes de la *Skip Connection* de la capa l.

En la figura 3.8 vemos como actua el modelo de antención en el modelo *P2PaLA*. g_l es la salida de la capa anterior del *decoder*, mientras que x_l es la *Skip connection*, que proviene del mismo nivel de del *encoder*. Lo que hace es transformar la entrada proveniente del *Skip Connection* del nivel l, x_l, en otra entrada ponderada por los pesos del model: $\hat{x}^l = \alpha^l x^l$.

3.4.4. *Res Path*

En [Ibtehaz and Rahman 2019] se muestra una forma de mejorar las *Skip Connections* añadiendo convoluciones en éstas. Se parte de la hipótesis de que la fusión entre las características del *encoder* y el *decoder* a cada nivel puede crear ciertas discrepancias en el aprendizaje y, debido a ello, en las predicciones. Esta discrepancia se va aliviando a medida que se mueven a través de la red, es decir, a medida que *encoder* y *decoder* se acercan. A niveles con mayor resolución de la imagen estas discrepancias son mínimas.

Para disminuir estas discrepancias se propone añadir una serie de convoluciones a las *Skip Connections*. A estas le ponemos el nombre de *Res Path*, compuestos por bloques de una convolución con *kernel* de 3×3 junto a otra convolución con *kernel* de 1×1, ambas sin cambios de tamaño en la imagen de entrada, con función de activación *ReLU*, *batch norm* y sumando el resultado. Se concatenan varios bloques seguidos (n bloques), dependiendo del nivel de la capa en el que se encuentre, como se puede ver en la imagen 3.9. El resultado final del bloque *Res Path* es modificar la *Skip Connection* x_l.

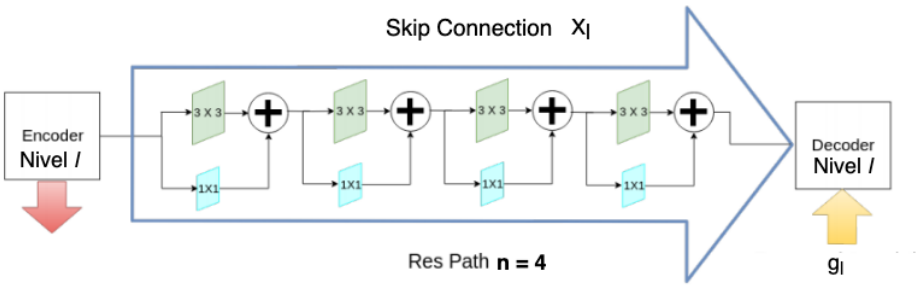


Figura 3.9: *Res Path*. Método propuesto de convoluciones en las *Skip Connections*. Imagen originalmente extraída de [Ibtehaz and Rahman 2019] y modificada.

Para este trabajo hemos optado, después de realizar algunas pruebas preliminares, por usar los bloques *Res Path* en las *Skip Connections* de los primeros cincos niveles. Hemos empezado añadiendo al primer nivel un bloque *Res Path* con $n = 6$, es decir, 6 bloques de convoluciones 3×3 y 1×1, y disminuido n en uno a cada nivel. Como resultado tenemos $6 * 2$ convoluciones en la *Skip Connection* en el primer nivel, $5 * 2$ convoluciones en la *Skip Connection* del segundo nivel, $4 * 2$ convoluciones en la *Skip Connection* del tercer nivel, y así hasta el quinto nivel, donde no se han añadido más.

En la figura 3.10 vemos nuestra implementación *Res Path* en *P2PaLa*.

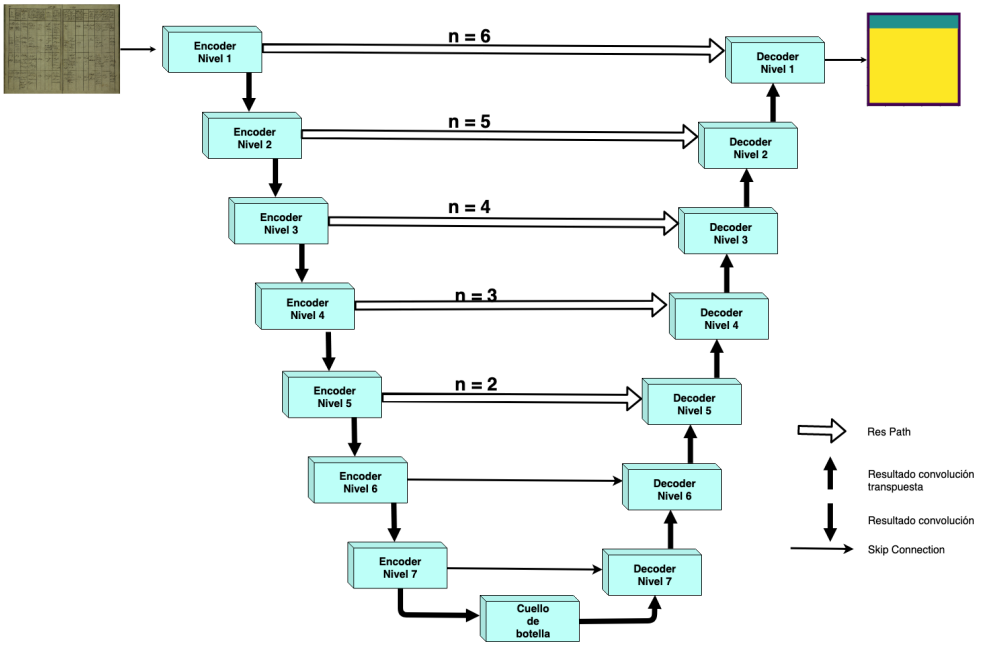


Figura 3.10: *Res Path* implementado en *P2PaLa*

3.4.5. *Nested U-Net*

Esta arquitectura no está basada en *P2PaLA*. En [Zhou et al. 2018] se presenta una arquitectura llamada *Nested U-Net*, la cual aplica, también, ciertas mejoras en las *Skip Connections*.

Se trata de una red basada en la arquitectura de la *U-Net* original con 5 niveles. En cada *Skip Connection* de cada nivel, se añaden una serie de convoluciones con el resultado del *encoder* de ese nivel y el resultado del *encoder* en un nivel inferior, realizando un *upsampling* para que el tamaño de la imagen de entrada concuerde. Se muestra el calculo de una convolución en la ecuación 3.9, donde $X^{i,j}$ es la convolución j del nivel i, $\Phi(\cdot)$ es la función de activación, U es la operación de *Up-Sampling*, $K^{i,j}$ es el *kernel* de la convolución en la capa i, j y los corchetes representan la concatenación. A cada convolución se le concatena no solo el resultado de la salida del *encoder* del mismo nivel sino también el resultado de las convoluciones anteriores del mismo nivel del *Skip Connection*. Esto último aumenta el número de parámetros de las convoluciones a medida que se aumenta la profundidad de la red.

$$X^{i,j} = \Phi([X^{i,0}, ..., X^{i,j-1}, U(X^{i+1,j-1})] * K^{i,j}) \tag{3.9}$$

En la figura 3.11 se puede observar la arquitectura. $X^{0,0}$ es la entrada de la red y $X^{0,4}$ la salida. El primer superíndice indica el nivel de profundidad de la arquitectura y el segundo indica el nivel de anchura, el cual depende del número de convoluciones en las *Skip Connections*. Por ejemplo, la entrada de la tercera convolución del primer nivel ($X^{0,2}$) sería la concatenación de las dos convoluciones anteriores del mismo nivel ($X^{0,0}, X^{0,1}$) junto al *Up-Sampling* de la convolución anterior en un nivel inferior ($X^{1,1}$), tras usar la función de activación $\Phi(\cdot)$: $X^{0,2} = \Phi(K^{0,2} * [X^{0,0}, X^{0,1}, U(X^{1,1})])$

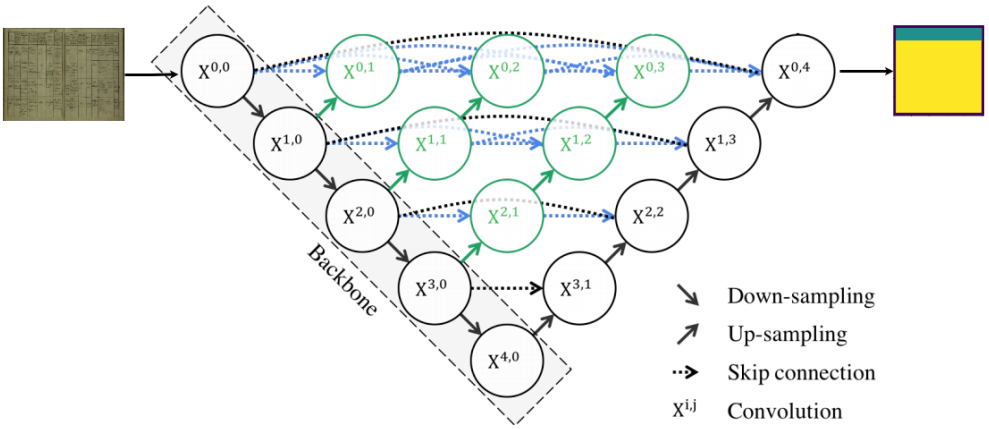


Figura 3.11: *Nested U-Net. Nested U-Net*, *encoder* y *decoder* conectados a través de una serie de convoluciones totalmente conectadas entre si. Imagen originalmente extraída de [Zhou et al. 2018] y posteriormente modificada para este caso.

Al estar todas las convoluciones conectadas en un mismo nivel se convierte en una red convolucional totalmente conectada (*feed-forward fully connected convolutional network*). Dicha característica convierte la arquitectura en una *red de redes U-Net*, creando cuatro redes diferentes, una por *Skip connection*, y al ser todos los resultados posteriormente concatenados la red puede escoger el mejor modelo.

Esta definición de *red de redes U-Net* se puede ver de forma gráfica en la figura 3.12. En rojo se puede ver, con tres bloques de convoluciones, una *U-Net* pequeña. Si juntamos la roja con la parte verde se observa otra de mayor tamaño, en azul y amarillo otras dos de mayor tamaño cada una. Cada una de estas sub-redes comparten bloques con las redes anteriores de menor tamaño. El *encoder* de la red amarilla, formado por cuatro convoluciones, recoge al resto de *encoders* y al resto de *decoders*. Lo mismo pasa con el resto de las sub-redes y sus predecesoras.

En esta red no se hace uso de convoluciones transpuestas para realizar el *Up-sampling*, es decir, para aumentar de tamaño el tensor, sino que se realiza interpolación bilineal. Para realizar el *Down-sampling*, a diferencia del artículo original, se hace uso de convoluciones con un *stride* igual a 2 y un *kernel* igual a 4×4. Esto es debido a que después de realizar varias pruebas previas se demostró que el uso de estas si mejora respecto a un *Down-sampling* tradicional. En el caso del *Up-sampling*, el uso de convoluciones transpuestas no ha demostrado una clara mejora en dichas pruebas, por lo que se sigue haciendo uso del interpolación bilineal.

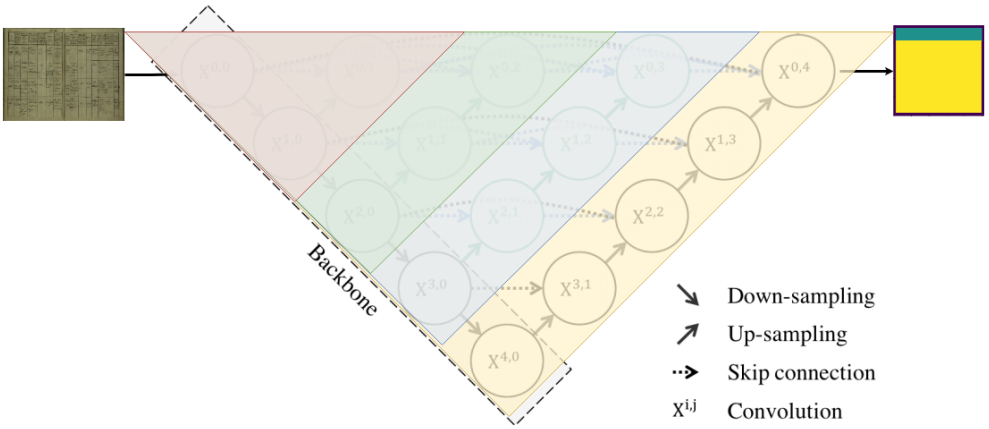


Figura 3.12: Diferentes *U-Nets* dentro de la *Nested U-Net*. En cada color se puede observar una arquitectura similar a la de una *U-Net*. Todas juntas forman una.

Dicha red es más compleja de entrenar, consumiendo también más memoria, teniendo que reducir drásticamente el *batch size* en la mayoría de casos. No obstate, ha conseguido los mejores resultados en una de las tareas propuestas como objetivo.

3.4.6. Métodos de fusión

Para tratar de aprovechar la información proveniente de las *class images* se han implementado dos métodos de fusión de la información proveniente de las *CI* y la imagen: *Early fusion* y *Middle fusion*.

En *Early fusion* hemos concatenado a la imagen original –la cual tiene tres canales RGB –un canal más por *CI*. Si se quiere aprovechar la información contextual de las palabras de cabecera, se crearán estas *CI* y se concatenarán a la imagen original, pasando a tener de tres a cuatro dimension.

En *Middle Fusion* se ha añadido a la red base usando un bloque formado por convoluciones con solo dichas *class images* como entrada, extrayendo la información necesaria de dichas imágenes.

Este método se ha probado con todas las redes anteriormente descritas, concatenando la información de estos nuevos bloques a la salida del modelo utilizado. El modelo utilizado previamente se modifica ligeramente en la última capa, haciendo que en lugar de devolver una capa de probabilidades devuelva 64 filtros.

El bloque convolucional para el caso de las *class images* es el siguiente:

$$A \rightarrow [C64, BN, ReLU] \rightarrow [C64, BN, ReLU] \rightarrow [C64, BN, ReLU]$$

Siendo A las *class images* como entrada, CX una convolución con un *kernel* de 3×3 sin variar el tamaño de la imagen y con X canales de salida. BN indica el uso de *Batch normalization*. Una vez se aplican las dos primeras

convoluciones, se concatena el resultado con el proveniente del modelo utilizado, se aplican dos convoluciones similares más y se extrae una capa de probabilidades.

En la figura 3.13 observamos el diseño de *Middle Fusion*.El modelo puede ser una de las arquitecturas anteriormente explicas: *P2PaLa*, *P2PaLa* con *Res Path*, *P2paLa* con *Squeeze Attention* o *Nested U-Net*. En la parte inferior se observa el bloque convoluciona lexplicado anteriormente. Se sigue con una concatenación del resultado del modelo y dicho bloque convolucional, seguido de otro bloque convolucional con el resulado de dicha concatenación.

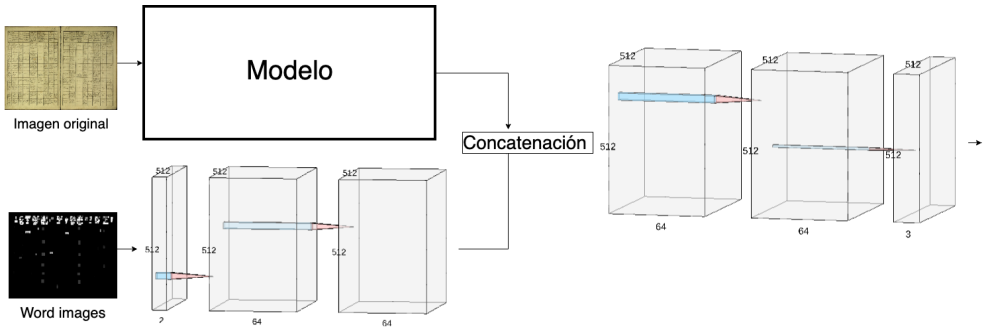


Figura 3.13: Modelo de *Middle fusion*

CAPÍTULO 4
Experimentos y resultados

En este capítulo se muestran los resultados de las dos tareas propuestas. Primero se mostrarán y explicarán los corpus utilizados. Después, se mostraran los resultados de la búsqueda de regiones en tablas y, seguidamente, de la separación entre actos.

4.1 Corpus

En esta sección se explica el corpus para cada tarea, mostrando su origen y utilidas, así como las particiones para cada tarea.

4.1.1. Passau

Este corpus, recopilado por los *Archivos Diocesanos de Passau*, contiene información sobre los feligreses que fueron bautizados, quienes se casaron y murieron dentro de los límites geográficos de las diversas parroquias de la diócesis de Passau. Las exploraciones se originan en más de 100 distritos pastorales con sus propios mantenimientos de registros, que comenzaron a finales del siglo XVI por la orden de la Iglesia. El corpus contiene más de 800,000 imágenes sacramentales. Este muestra la evolución de varios tipos de registro en el tiempo, demostrando una gran variedad en la escritura a mano, el namtenimiento de los registros y más formas de tablas estandarizadas introducidas a principios del siglo XIX. En la figura 4.1 se observan algunos ejemplos.

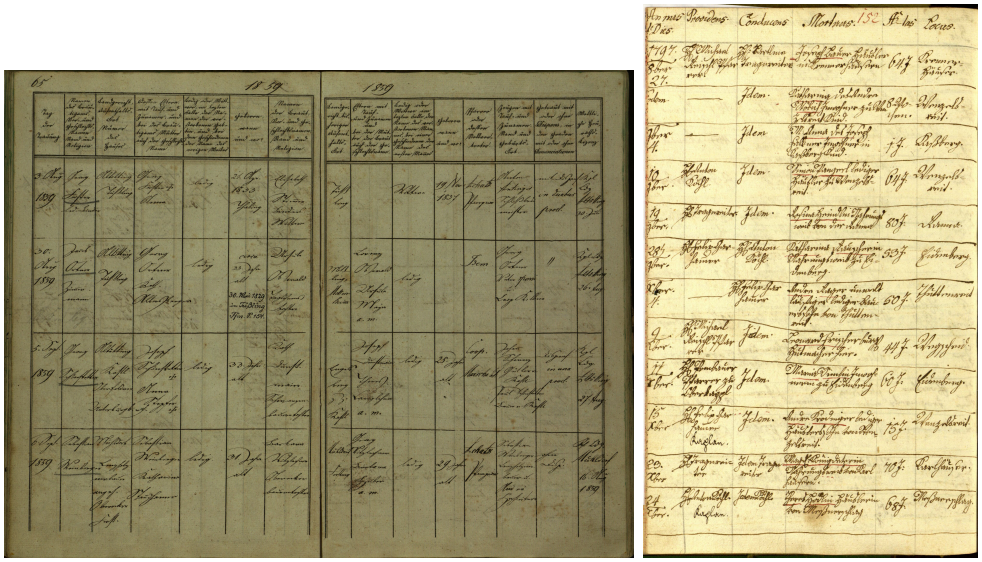


Figura 4.1: Ejemplo de páginas del corpus de *Passau*

Cada registro en los libros se refiere a un evento sacramental y, por lo tanto, proporciona referencias sobre la persona que fue batutizada en un dia determinado, una boda o un fallecimiento. Además de los nombres, referencias del puesto de trabajo y lugar donde vivía el individuo, entre otra mucha información. Por todo esto, los libros de registro no son sólo de interés para investigadores de la familia, que exploran su historia personal, sino que también demuestran la historia social, económica o humanitaria.

Se tiene acceso a un corpus reducido respecto al original, compuesto mayormente por tablas, compuesto por un total de 286 imágenes. Se ha dividido el corpus en 176 imágenes para el conjunto entrenamiento, 21 imágenes para el conjunto de desarrollo y 89 imágenes para el conjunto de test, con el que se sacarán los resultados.

Dado que para detectar las regiones (cabecera y contenido) no es necesario una gran resolución, las imágenes de este conjunto se re-dimensionarán a una resolución de 512×512 píxeles. Esto permitirá aumentar el número de imágenes que se usa en cada ciclo de la ANN, disminuyendo así el tiempo de entrenamiento e inferencia.

4.1.2. Chancery

El corpus completo, llamado *Chancery*, está compuesto por unas 70,000 imágenes, reproduciendo la colección de registros medievales. Estos registros fueron producidos por la cancillería real francesa (París, archivos nacionales, JJ 35 - JJ 211), en el periodo comprendido entre los años 1302 a 1483. Estos contienen 68,000 cartas dadas por el rey de Francia. Esta larga e icónica colección es testigo de la administración de la Edad Media. Es considerada una de las claves para ayudar a entender a la Europa Medieval y el auge de los estados como naciones centralizadas, apareciendo este auge en todo el continente como consecuencia de las duraderas guerras entre Francia e Inglaterra.

Figura 4.2: Ejemplo de páginas del corpus de *Chancery*

El tamaño de este corpus impide a los expertos estudiarlo tan exhaustivamente como se merece, por lo que un acceso fácil a los contenidos de estos recursos clave podría ayudar a aumentar el conocimiento de la historia medieval y, así, promover la investigación en estudios comparativos entre la gestión y la administración del estado.

Desde la perspectiva de la maquetación del corpus, está dividido entre *Actos* o *Cartas*. En estos *actos* se definieron las declaraciones dictadas por el soberano actual sobre la creación de diferentes organismos organizacionales, así como sus derechos y privilegios.

Los *actos* pueden ser suficientemente grandes como para estar distribuidos en diferentes páginas. Por esto, es importante para los investigadores determinar cuando empieza y termina cada uno. Si los *actos* estan correctamente delimitados los investigadores pueden buscar sobre ciertos actos o cartas que contengan información sorbe un tema específico.

Para este trabajo se ha accedido a un total de 556 imágenes correctamente etiquetadas sobre el corpus de *Chancery*. Se ha divido el corpus en 136 imágenes para el conjunto de entrenamiento, 19 imágenes para el conjunto de desarrollo y 411 imágenes para el conjunto de muestra, con el que se calcularán los resultados.

Existe un trabajo previo en la separación de *actos* [Bosch] pero en este se plantea el problema de una forma diferente. En el trabajo original se plantea la búsqueda de la separación de actos buscando no solo cuando acaba un acto, sino el inicio y fin del acto. En caso de que el acto esté en una sola página se describe como *acto completo* y en caso de que sea la continuación de un acto, sin llegar a acabarlo, se describe como *continuación de acto*.

En [Bosch] se tienen, en total, cuatro clases, dado que se considera que las páginas pueden llegar no ordenadas y, con el orden dado por las clases, se pueden llegar a ordenar los actos con un coste menor. En este trabajo se consideran dos clases: fin o continuación de acto. Esto es así porque el

corpus que actualmente se tiene está ordenado y no haría falta una ordenación posterior, sino solo separar dichos actos. Esto provoca que los resultados de ambos trabajos no sean comparables.

Dado que la división de los actos se tratará como una línea, se han redimensionado las imágenes a 1024×768, tamaño comúnmente usado en la extracción de líneas. Se ha probado de usar dimensiones más pequeñas dando pésimos resultados preliminares.

4.2 Tablas PASSAU

Se han ejecutado todas las arquitecturas de la sección 3.4 tres veces y se ha escogido el mejor resultado obtenido en cada una de ellas. Cada una se ha entrenado con todas las combinaciones posibles: con solo la imagen, estimando la distribución $\boldsymbol{P(y|x)}$, con las *class images* de palabras de cabecera, estimando la distribución $\boldsymbol{P(y|x, w_1)}$ y, finalmente, con las dos *class images* con las palabras de cabecera como de contenido, estimando $\boldsymbol{P(y|x, w_1, w_2)}$, siendo w_1 y w_2 las *class images* con las palabras de cabecera y de contenido respectivamente, y x la imagen de entrada.

Para la arquitectura *P2PaLA* se ha usado un tamaño de *batch size* de doce imágenes excepto en el caso de *middle fusion*, el cual se ha disminuido a seis. Con los modelos *Squeeze attention* y *Res Path* se han usado un *batch size* de seis imágenes mientras que usando *middle fusion* se ha disminuido a cinco. Con el modelo *Nested*, el más demandante en cuestiones de memoria y tiempo, se ha utilizado un *batch size* de cuatro y se ha disminuido a tres al usar *middle fusion*. Dichos tamaños de *mini batch* se han establecido por la cantidad de memoria necesaria de cada arquitectura y la disponible.

Tabla 4.1: Resultados de *mIoU* en corpus de *PASSAU*. [1]

		$\mathrm{P}(\boldsymbol{y}\|\boldsymbol{x}, \boldsymbol{w_1})$		$\mathrm{P}(\boldsymbol{y}\|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$	
	$\mathrm{P}(\boldsymbol{y}\|\boldsymbol{x})$	**EF**	**MF**	**EF**	**MF**
P2PaLA	$75{,}5 \pm 5{,}9$	$70{,}8 \pm 6{,}4$	$80{,}6 \pm 5{,}1$	$59{,}5 \pm 5{,}1$	$78{,}8 \pm 4{,}6$
Squeeze Att.	$78{,}0 \pm 5{,}5$	$74{,}0 \pm 6{,}0$	$73{,}3 \pm 6{,}3$	$67{,}1 \pm 5{,}3$	$67{,}9 \pm 4{,}9$
Res Path	$78{,}1 \pm 5{,}6$	$75{,}2 \pm 6{,}1$	$72{,}3 \pm 5{,}9$	$61{,}8 \pm 5{,}0$	$69{,}7 \pm 4{,}6$
Nested	$\mathbf{83{,}8 \pm 4{,}3}$	$78{,}0 \pm 5{,}2$	$78{,}8 \pm 4{,}5$	$53{,}4 \pm 3{,}1$	$79{,}0 \pm 4{,}14$

En la tabla 4.1 vemos los resultados de todas las operaciones. Se usa la métrica *mIoU* estimando las probabilidades $P(\boldsymbol{y}|\boldsymbol{x})$, $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1})$ y $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$. Cuando se usan las dimensiones w_1 y w_2, es decir, las *class images* de palabras de cabecera y contenido, respectivamente, se trata de usar *early fusion* (EF) concatenando en la propia imagen las *class images* como entrada para la red (pasa de tener tres canales de entrada a tener cuatro y cinco), y usando *middle fusion* (MF), modelo explicado en la sub-sección 3.4.6.

Podemos ver que usando solo la imagen original, estimando $P(\boldsymbol{y}|\boldsymbol{x})$, los resultados de *P2PaLA*, los cuales usamos como línea base de resultados, obtenemos un *mIoU* de 75,5. Con el resto de modelos se consigue mejorar los resultados. Con *Squeeze attention* los resultados mejoran en 2,5 puntos y con

[1] Los intervalos de confianza se han calculado mediante *bootstrap* con 20,000 iteraciones al 95 %. Para mejorar la legibilidad se muestra el valor más pesimista debido al tamaño de la tabla.

Res Path mejoran en 2,6. Con la arquitectura *Nested* el modelo mejora en **8,3** puntos, lo que es la mejora más grande y significativa de todas.

Si nos fijamos en los resultados de *EF*, estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1})$ y $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$, los resultados no mejoran, siendo incluso peores. A mayor información en los canales de entrada más baja la métrica, como se puede observar con $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$ respecto a $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1})$, los cuales empeoran, y estos últimos respecto a $P(\boldsymbol{y}|\boldsymbol{x})$ también. Se entiende de esto que el modelo no es capaz de aprender a fusionar la información y solo crea ruido en la entrada, el cual no es capaz de eliminar.

Si analizamos los resultados del método *MF* vemos que la arquitectura de *P2PaLA* ha aprovechado la información de las *class images* mejorando los resultados con ambas imágenes. Estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1})$ se obtiene un *mIoU* de 80,6, mejorando en 5,1 puntos la base de la propia arquitectura. Estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$ con *MF* se mejora en 3,3 puntos la base pero siendo menor la mejora que usando solo una dimensión. En cambio las arquitecturas *Res Path y Squeeze attention* no mejoran con *MF*, sin aprovechar la información de las *class images* incluso empeorando los resultados previos. *Nested*, arquitectura con el mejor resultado sin fusión, tampoco aprovecha la informacion de las *class images*, empeorando drásticamente al estimar $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$.

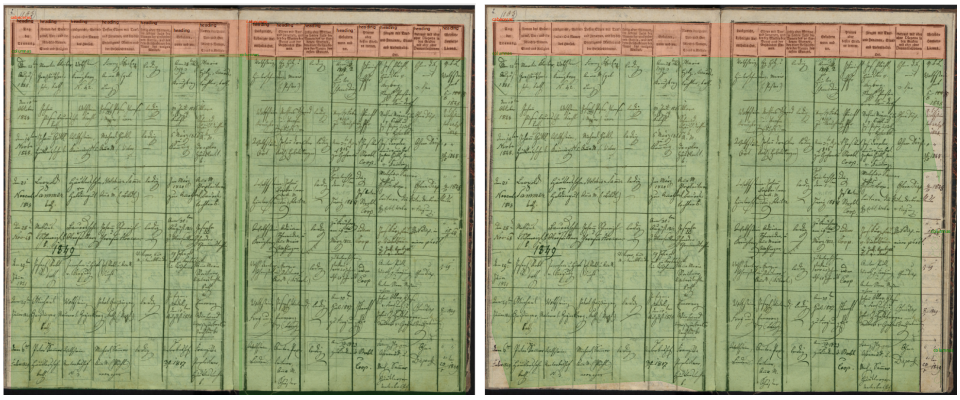


Figura 4.3: Resultados en corpus de *PASSAU*. A la izquierda la página etiquetada y a la derecha la hipótesis. Resultado con el modelo *Nested* sin *class images*.

En la figura 4.3 vemos el resultado de la maquetación de una página con una tabla bien definida. A la izquierda la página correctamente etiquetada y a la derecha el resultado del método *Nested* sin *class images*, el cual ha dado los mejores resultados. En esta página se detecta bien la cabecera (rojo sombreado), fallando muy poco, mientras que el contenido de la tabla (verde sombreado) también se acierta aunque menos, al fallar más en los bordes. El modelo ha fallado en otras tablas, tal y como vemos en la figura 4.4, en la cual no ha conseguido unificar el contenido y apenas ha acertado con la cabecera.

Figura 4.4: Fallos en corpus de *PASSAU*. A la izquierda la página etiquetada y a la derecha la hipótesis. Resultado con el modelo *Nested* sin *class images*.

Podemos analizar los tiempos de ejecución de cada arquitectura en la tabla 4.2. Observamos que la arquitectura más rápida en todas las pruebas es *P2PaLA* seguida de *Squeeze attention* con poca diferencia entre ellas. La arquitectura *Nested* es la más lenta con diferencia, como también veíamos con el tamaño del *batch size*.

Tabla 4.2: Tiempo de entrenamiento de los diferentes modelos en corpus de *PASSAU*. Tiempo en minutos.

		$P(\boldsymbol{y}\|\boldsymbol{x},\boldsymbol{w_1})$		$P(\boldsymbol{y}\|\boldsymbol{x},\boldsymbol{w_1},\boldsymbol{w_2})$	
	$P(\boldsymbol{y}\|\boldsymbol{x})$	**EF**	**MF**	**EF**	**MF**
P2PaLA	74.3	78.9	136.5	87.5	137.8
Squeeze Att.	80.4	87.3	135.3	91.1	138.2
Res Path	132.6	134.6	188.1	136.7	189.9
Nested	436.7	437.2	493.2	440.2	492.3

4.3 Chancery

En este problema se han ejecutado todas las arquitecturas de la sección 3.4 excepto *Nested*, debido a que el tamaño de entrada en este caso es de 1024×768 y dicha arquitectura demanda demasiado memoria como para ejecutar con imágenes de tal tamaño, por lo que se ha descartado su uso en este problema.

Para la arquitectura *P2PaLA* se ha usado un tamaño de *batch size* de ocho imágenes excepto en el caso de *middle fusion*, el cual se ha disminuido a dos. Con los modelos *Squeeze attention* y *Res Path* se han usado un *batch size* de ocho y tres imágenes, respectivamente, mientras que usando *middle fusion* se ha disminuido a una imagen en ambos casos.

Una vez extraidas las líneas, se realiza un post-proceso sobre estas específico para este problema. Se han eliminado las líneas de longitud menor al setenta

por ciento de la página y alargado el resto de líneas para que completen toda la longitud de esta. Esto es debido a que en el corpus etiquetado todas las separaciones de acto van de lado a lado de la página siempre y eliminar estas líneas es equivalente a eliminar parte del ruido que la red produce.

Una vez realizado el corte y alargamiento de líneas, se realiza, mediante técnicas de *clustering*, la unión de líneas con poca distancia entre ellas, dado que a veces se observan más de una separación diferenciada por algunos píxeles, debido al anterior alargamiento de las líneas y dado de que no disponemos una probabilidad a priori de quan largo es un acto, aunque si observamos el corpus veremos que nunca es de unas pocas líneas. Se ha utilizado *K-means*, probando con diferente número de clústers, siendo cada uno una línea. Cuando se encuentra una suma de la distancia de los clúster al cuadrado que supere cierto umbral, se unirán líneas con ese número de clústers. Dicho umbral ha sido establecido en 10,000 en este corpus, encontrado al realizar varias pruebas con diferentes umbrales.

Tabla 4.3: Resultados de *F1-measure* en *Chancery*. [2]

		$P(\boldsymbol{y}\|\boldsymbol{x},\boldsymbol{w_1})$		$P(\boldsymbol{y}\|\boldsymbol{x},\boldsymbol{w_1},\boldsymbol{w_2})$	
	$P(\boldsymbol{y}\|\boldsymbol{x})$	**EF**	**MF**	**EF**	**MF**
P2PaLA	$79{,}1 \pm 3{,}5$	$75{,}3 \pm 3{,}7$	$81{,}7 \pm 3{,}0$	$78{,}9 \pm 3{,}2$	$\mathbf{83{,}5 \pm 2{,}8}$
Squeeze Att.	$77{,}2 \pm 3{,}4$	$77{,}8 \pm 3{,}5$	$80{,}1 \pm 3{,}1$	$74{,}5 \pm 3{,}7$	$81{,}3 \pm 2{,}9$
Res Path	$74{,}8 \pm 3{,}4$	$78{,}4 \pm 3{,}3$	$78{,}3 \pm 3{,}3$	$80{,}2 \pm 2{,}9$	$83{,}5 \pm 2{,}8$

Se muestran los resultados de la separación de los actos en la tabla 4.3. De forma similar al apartado anterior, se han estimado las probabilidades $P(\boldsymbol{y}|\boldsymbol{x})$, $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1})$ y $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1},\boldsymbol{w_2})$, usando las dimensionees w_1 y w_2 como *class images* de palabras de fin y de inicio de acto, respectivamente. En la tabla aparece *early fusion* como EF y *middle fusion* como MF, explicado en la subsección 3.4.6.

Utilizando solo la imagen original ($P(\boldsymbol{y}|\boldsymbol{x})$), los resultados de *P2PaLA*, usados como línea base de resultados, alcanzamos un *F1* de 79,11. Con los otros dos modelos, *Squeeze attention* y *Res Path* no se alcanzan resultados mejores.

Empleando EF, estimando $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1})$, los resultados no han mejorado en ningún momento a los resultados base, por lo que ningún modelo ha podido aprovechar la información de las palabras de fin de actos. En cambio, utilizando EF y estimando $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1},\boldsymbol{w_2})$, el modelo *Res Path* mejora en casi un punto respecto a la línea base, aprovechando la información de ambas dimensiones añadidas, alcanzando 80,20 en *F1*.

Con MF se han aprovechado mejor la información que ofrecen las *class images*. Estimando $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1})$, con las palabras de fin de acto, se ha alcanzado 81,73 en *F1*, mejorando en 2,68 puntos respecto a la línea base. Si estimamos $P(\boldsymbol{y}|\boldsymbol{x},\boldsymbol{w_1},\boldsymbol{w_2})$, utilizando tanto las palabras de fin de acto como las de inicio, se alcanza el mejor resultado, con 83,53 en *F1*, mejorando en 4,42 puntos a la línea base. Los mejores modelos se han encontrado con el modelo base de *P2PaLA*.

[2] Los intervalos de confianza se han calculado mediante *bootstrap* con 20,000 iteraciones al 95 %. Para mejorar la legibilidad se muestra el valor más pesimista debido al tamaño de la tabla.

Cada una de las arquitecturas ha mejorado usando MF, alcanzando el mejor resultado usando ambas dimensiones extra añadidas, estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$ y obteniendo mejoras significativas sobre los resultados base, por lo que se ha podido aprovechar, en este problema, la información añadida en base a las palabras.

Figura 4.5: Resultados de separación de actos en corpus de *Chancery*. A la izquierda la página etiquetada y a la derecha la hipótesis. Resultado con el modelo *P2PaLA*, estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$ usando MF.

En la figura 4.5 vemos una separación de cuatro actos. Al separar los trescientas primeros actos se obtienen muy buenos resultados, estamos las líneas de separación muy a la par. En cambio, al separar el tercer del cuarto acto (en la página siguiente seguiría este cuarto acto), la línea difiere un poco a la original.

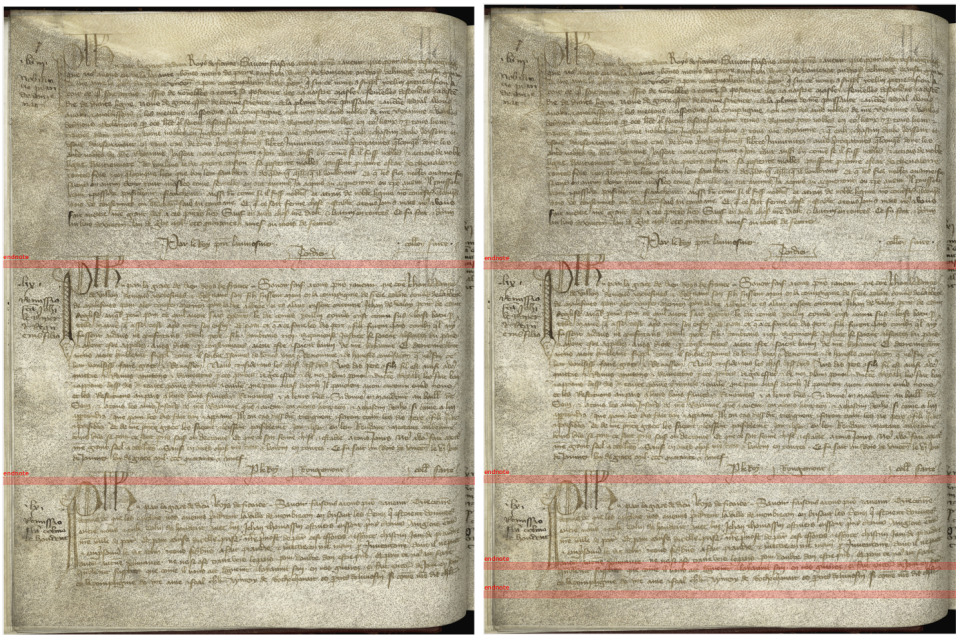


Figura 4.6: Fallos en los resultados de separación de actos en corpus de *Chancery*. A la izquierda la página etiquetada y a la derecha la hipótesis. Resultado con el modelo *P2PaLA*, estimando $P(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$ usando MF.

Si observamos otras imágenes se observa como el modelo falla más al intentar separar actos habitualmente al final de la página, tal y como vemos en la figura 4.6. En esta imagen se intenta separar la página en cinco actos, cuando realmente hay tres, lo que disminuye el *recall*.

Analizamos los tiempos de entrenamiento para cada modelo después de 300 *epochs*, anotados en la tabla 4.4. Observamos como con el modelo más básico, *P2PaLA*, se consiguen los tiempos más bajos al igual que el mejor resultado, tal y como hemos visto en tabla 4.3. Al utilizar MF los tiempos aumentan debido a las capas convolucionales de alta dimensionalidad añadidas. Si lo unimos a la arquitectura *Res Path* lo tiempos se disparan hasta casi los 800 minutos de entrenamiento, sin llegar a obtener mejores resultados. El mejor modelo anotado en la tabla 4.3 se obtiene tras **304,21** minutos de entrenamiento.

Tabla 4.4: Tiempo de entrenamiento de los diferentes modelos en corpus de *Chancery*. Tiempo en minutos.

		$P(\boldsymbol{y}\|\boldsymbol{x}, \boldsymbol{w_1})$		$P(\boldsymbol{y}\|\boldsymbol{x}, \boldsymbol{w_1}, \boldsymbol{w_2})$	
	$P(\boldsymbol{y}\|\boldsymbol{x})$	**EF**	**MF**	**EF**	**MF**
P2PaLA	111.58	120.43	301.05	128.65	304.21
Squeeze Att.	118.85	125.95	392.50	133.33	394.00
Res Path	225.28	223.81	792.81	234.35	792.88

CAPÍTULO 5
Conclusiones

Cabe recordar en un primer momento que en este proyecto se habían planteado dos objetivos principales. El primero era la exploración de diversas alternativas de arquitecturas de redes neuronales convolucionales y el segundo, era el análisis del impacto del uso de índices probabilísticos en problemas de maquetación. Para ello, se habían marcado una serie de metas sobre las cuales podemos sacar una serie de conclusiones.

En base a la primera meta, se ha obtenido un resultado base en cada tarea con la arquitectura *P2PaLA*. Se ha re-implementado dicha arquitectura para prepararla para las modificaciones del propio proyecto.

En la segunda meta, la cual se basaba en explorar diferentes arquitecturas, se han encontrado mejoras en la tarea de detección de zonas en tablas. Para encontrar dichas mejoras, se han probado hasta tres arquitecturas distintas y cada una ha mejorado los resultados de la anterior, siendo la arquitectura *Nested U-Net* la que obtenido mejores resultados. En cuanto a la separación de actos, no se han observado mejoras con dichas arquitecturas.

La tercera meta se centraba en analizar el impacto de los índices probabilísticos. Para ello, se han creado las llamadas *class images*, imágenes que representan dichos índices con las palabras que más nos interesan a fin de poder fusionar la información con la que nos ofrecen las propias imágenes. Pese a no conseguir buenos resultados con el método de *Early Fusion*, el método de *Middle Fusion* si logra mejoras. Estas mejoras alcanzan a ser hasta más de 4 puntos en ambas tareas dependiendo de la arquitectura utilizada.

Y por último, la cuarta meta se ha basado en analizar el impacto de los índices probabilísticos en las diferentes arquitecturas probadas. En cuanto a la detección de zonas en tablas, las nuevas arquitecturas no han sido capaces de extraer la nueva información de forma correcta. Por el contrario, la arquitectura base *P2PaLA* si lo ha hecho. En cuanto a la separación de actos, todas las arquitecturas han conseguido extraer la información y han mejorado sus resultados utlizando la técnica de *Middle Fusion*, logrando alcanzar el mejor resultado con el modelo base *P2PaLA*.

Visto lo anterior, cabe concluir que no hay una arquitectura ideal para mejorar los resultados en estas dos tareas al mismo tiempo. Al utlizar *Early Fusion* los resultados empeoran, mientras que con *Middle Fusion* los resultados mejoran en varios casos. En ambas tareas se han conseguido mejorar los resultados base.

5.1 Trabajos futuros

Como líneas de investigación futuras se presentan las siguientes opciones:

- Realización de pruebas más robustas para corroborar los resultados, las cuales requieren más tiempo de cómputo.
- Mejorar la creación de las *class images* mediante la creación de un mejor clasificador de palabras, teniendo en cuenta el corpus y basándola en métodos más eficientes que la frecuencia de las palabras.
- Explorar el uso de *Word Embeddings* para la representación de palabras en imágenes.
- Analizar otras técnicas de fusión como puede ser *Late Fusion*.
- Explorar nuevas funciones de pérdida para los problemas de segmentación, tales como DICE o Lovasz-Softmax [Berman et al. 2018].

Referencias

[Ares-Oliveira et al. 2018] Ares-Oliveira, S., Seguin, B., and Kaplan, F. (2018). DhSegment: A generic deep-learning approach for document segmentation. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2018-Augus:7–12.

[Baechler and Ingold 2011] Baechler, M. and Ingold, R. (2011). Multi resolution layout analysis of medieval manuscripts using dynamic MLP. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 1185–1189.

[Berman et al. 2018] Berman, M., Triki, A. R., and Blaschko, M. B. (2018). The Lovasz-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 4413–4421.

[Bishop 2006] Bishop, C. (2006). *Pattern Recognition and Machine Learning*. Springer-Verlag New York,, Secaucus, NJ, USA.

[Bluche et al. 2017] Bluche, T., Hamel, S., Kermorvant, C., Puigcerver, J., Stutzmann, D., Toselli, A. H., and Vidal, E. (2017). Preparatory KWS Experiments for Large-Scale Indexing of a Vast Medieval Manuscript Collection in the HIMANIS Project. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, 1:311–316.

[Bosch] Bosch, V. *Advances In Document Layout Analysis*. PhD thesis, Universitat Politècnica de València.

[Bukhari et al. 2012] Bukhari, S. S., Breuel, T. M., Asi, A., and El-Sana, J. (2012). Layout analysis for Arabic historical document images using machine learning. *Proceedings - International Workshop on Frontiers in Handwriting Recognition, IWFHR*, pages 639–644.

[Diederik P. and Ba 2014] Diederik P., K. and Ba, L. (2014). Adam: A Method for Stochastic Optimization. *AIP Conference Proceedings*, 1631:58–62.

[Eskenazi et al. 2017] Eskenazi, S., Gomez-Krämer, P., and Ogier, J. M. (2017). A comprehensive survey of mostly textual document segmentation algorithms since 2008. *Pattern Recognition*, 64:1–14.

[Fern and Terrades 2012] Fern, F. C. and Terrades, O. R. (2012). Document Segmentation using Relative Location Features. *International Conference on Pattern Recognition*, (Icpr):1562–1565.

[G 2000] G, N. (2000). Twenty Years of Document Image Analysis in PAMI. *IEEE Transactions on pattern analysis and machine intelligence*, 22(1).

[Goodfellow et al. 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[Gruning et al. 2018] Gruning, T., Labahn, R., Diem, M., Kleber, F., and Fiel, S. (2018). READ-BAD: A new dataset and evaluation scheme for baseline detection in archival documents. In *Proceedings - 13th IAPR International Workshop on Document Analysis Systems, DAS 2018*, pages 351–356.

[Grüning et al. 2018] Grüning, T., Leifert, G., Strauß, T., and Labahn, R. (2018). A Two-Stage Method for Text Line Detection in Historical Documents. pages 1–38.

[Hornik 1991] Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

[Ibtehaz and Rahman 2019] Ibtehaz, N. and Rahman, M. S. (2019). MultiResUNet : Rethinking the U-Net Architecture for Multimodal Biomedical Image Segmentation. pages 1–25.

[Ioffe and Szegedy 2015] Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift.

[Lang et al. 2018] Lang, E., Puigcerver, J., Toselli, A. H., and Vidal, E. (2018). Probabilistic indexing and search for information extraction on handwritten German parish records. *Proceedings of International Conference on Frontiers in Handwriting Recognition, ICFHR*, 2018-Augus:44–49.

[Long et al. 2018] Long, J., Shelhamer, E., and Darrel, T. (2018). Fully Convolutional Adaptation Networks for Semantic Segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 6810–6818.

[Oktay et al. 2018] Oktay, O., Schlemper, J., Folgoc, L. L., Lee, M., Heinrich, M., Misawa, K., Mori, K., McDonagh, S., Hammerla, N. Y., Kainz, B., Glocker, B., and Rueckert, D. (2018). Attention U-Net: Learning Where to Look for the Pancreas. (Midl).

[Puigcerver 2018] Puigcerver, J. (2018). *A Probabilistic Formulation of Keyword Spotting*. PhD thesis, Universitat Politècnica de València.

[Quirós 2018] Quirós, L. (2018). Multi-Task Handwritten Document Layout Analysis. pages 1–23.

[Ronneberger et al. 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9351:234–241.

[Roseblatt et al. 1957] Roseblatt, F., Stieber, A., and Shatz, R. H. (1957). The Perceptron: A perceiving and recognizing automaton. Technical report.

[Rumelhart et al. 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Backprop_Old.Pdf.

[Simard et al. 2003] Simard, P. Y., Steinkraus, D., and Platt, J. C. (2003). Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis. (ICDAR):1–6.

[Srivastava et al. 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[Toselli et al. 2010] Toselli, A. H., Romero, V., Pastor, M., and Vidal, E. (2010). Multimodal interactive transcription of text images. *Pattern Recognition*, 43(5):1814–1825.

[Vidal et al. 2019] Vidal, E., Toselli, A. H., and Puigcerver, J. (2019). A Probabilistic Framework for Lexicon-based Keyword Spotting in Handwritten Text Images. *arXiv, Tech. Rep.*

[Wei et al. 2013] Wei, H., Baechler, M., Slimane, F., and Ingold, R. (2013). Evaluation of SVM, MLP and GMM classifiers for layout analysis of historical documents. *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pages 1220–1224.

[Zhou et al. 2018] Zhou, Z., Rahman Siddiquee, M. M., Tajbakhsh, N., and Liang, J. (2018). Unet++: A nested u-net architecture for medical image segmentation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11045 LNCS:3–11.