Universitat Politècnica de València
Departamento de Sistemas Informáticos y Computación
Departamento de Comunicaciones

Contributions to Deep Learning Models

by

Jordi Mansanet Sandín

Thesis presented at Universitat Politècnica de València in partial fulfillment of the requirements for the degree of Doctor.

Supervisors:
Dr. Alberto Albiol
Dr. Roberto Paredes

Valencia, November 26th 2015

A Aisha,

Abstract / Resumen / Resum

Deep Learning is a new area of Machine Learning research which aims to create computational models that learn several representations of the data using deep architectures. These methods have become very popular over the last few years due to the remarkable results obtained in speech recognition, visual object recognition, object detection, natural language processing, etc.

The goal of this thesis is to present some contributions to the Deep Learning framework, particularly focused on computer vision problems dealing with images. These contributions can be summarized in two novel methods proposed: a new regularization technique for Restricted Boltzmann Machines called Mask Selective Regularization (MSR), and a powerful discriminative network called Local Deep Neural Network (Local-DNN). On the one hand, the MSR method is based on taking advantage of the benefits of the L_2 and the L_1 regularizations techniques. Both regularizations are applied dynamically on the parameters of the RBM according to the state of the model during training and the topology of the input space. On the other hand, the Local-DNN model is based on two key concepts: local features and deep architectures. Similar to the convolutional networks, the Local-DNN model learns from local regions in the input image using a deep neural network. The network aims to classify each local feature according to the label of the sample to which it belongs, and all of these local contributions are taken into account during testing using a simple voting scheme.

The methods proposed throughout the thesis have been evaluated in several experiments using various image datasets. The results obtained show the great performance of these approaches, particularly on gender recognition using face images, where the Local-DNN improves other state-of-the-art results.

El Aprendizaje Profundo (Deep Learning en inglés) es una nueva área dentro del campo del Aprendizaje Automático que pretende crear modelos computacionales que aprendan varias representaciones de los datos utilizando arquitecturas profundas. Este tipo de métodos ha ganado mucha popularidad durante los últimos años debido a los impresionantes resultados obtenidos en diferentes tareas como el reconocimiento automático del habla, el reconocimiento y la detección automática de objetos, el procesamiento de lenguajes naturales, etc.

El principal objetivo de esta tesis es aportar una serie de contribuciones realizadas dentro del marco del Aprendizaje Profundo, particularmente enfocadas a problemas relacionados con la visión por computador. Estas contribuciones se resumen en dos *novedosos métodos: una nueva técnica de regularización para Restricted Boltzmann Machines llamada Mask Selective Regularization (MSR), y una potente red neuronal discriminativa llamada Local Deep Neural Network (Local-DNN). Por una lado, el método MSR se basa en aprovechar las ventajas de las técnicas de regularización clásicas basadas en las normas L_2 y L_1. Ambas regularizaciones se aplican sobre los parámetros de la RBM teniendo en cuenta el estado del modelo durante el entrenamiento y la topología de los datos de entrada. Por otro lado, El modelo Local-DNN se basa en dos conceptos fundamentales: características locales y arquitecturas profundas. De forma similar a las redes convolucionales, Local-DNN restringe el aprendizaje a regiones locales de la imagen de entrada. La red neuronal pretende clasificar cada característica local con la etiqueta de la imagen a la que pertenece, y, finalmente, todas estas contribuciones se tienen en cuenta utilizando un sencillo sistema de votación durante la predicción.*

Los métodos propuestos a lo largo de la tesis han sido ampliamente evaluados en varios experimentos utilizando distintas bases de datos, principalmente en problemas de visión por computador. Los resultados obtenidos muestran el buen funcionamiento de dichos métodos, y sirven para validar las estrategias planteadas. Entre ellos, destacan los resultados obtenidos aplicando el modelo Local-DNN al problema del reconocimiento de género utilizando imágenes faciales, donde se han mejorado los resultados publicados del estado del arte.

L'Aprenentatge Profund (Deep Learning en anglès) és una nova àrea dins el camp de l'Aprenentatge Automàtic que pretén crear models computacionals que aprenguen diverses representacions de les dades utilitzant arquitectures profundes. Aquest tipus de mètodes ha guanyat molta popularitat durant els últims anys a causa dels impressionants resultats obtinguts en diverses tasques com el reconeixement automàtic de la parla, el reconeixement i la detecció automàtica d'objectes, el processament de llenguatges naturals, etc.

El principal objectiu d'aquesta tesi és aportar una sèrie de contribucions realitzades dins del marc de l'Aprenentatge Profund, particularment enfocades a problemes relacionats amb la visió per computador. Aquestes contribucions es resumeixen en dos nous mètodes: una nova tècnica de regularització per Restricted Boltzmann Machines anomenada Mask Selective Regularization (MSR), i una potent xarxa neuronal discriminativa anomenada Local Deep Neural Network (Local-DNN). D'una banda, el mètode MSR es basa en aprofitar els avantatges de les tècniques de regularització clàssiques basades en les normes L_2 i L_1. Les dues regularitzacions s'apliquen sobre els paràmetres de la RBM tenint en compte l'estat del model durant l'entrenament i la topologia de les dades d'entrada. D'altra banda, el model Local-DNN es basa en dos conceptes fonamentals: característiques locals i arquitectures profundes. De forma similar a les xarxes convolucionals, Local-DNN restringeix l'aprenentatge a regions locals de la imatge d'entrada. La xarxa neuronal pretén classificar cada característica local amb l'etiqueta de la imatge a la qual pertany, i, finalment, totes aquestes contribucions es fusionen durant la predicció utilitzant un senzill sistema de votació.

Els mètodes proposats al llarg de la tesi han estat àmpliament avaluats en diversos experiments utilitzant diferents bases de dades, principalment en problemes de visió per computador. Els resultats obtinguts mostren el bon funcionament d'aquests *mètodes, i serveixen per validar les estratègies plantejades. Entre d'ells, destaquen els resultats obtinguts aplicant el model Local- DNN al problema del reconeixement de gènere utilitzant imatges facials, on s'han millorat els resultats publicats de l'estat de l'art.*

Acknowledgments

I would like to thank many people who helped me along the path to writing this thesis. This work is not only the result of all my efforts but a consequence of the many supports that I have received.

To Alberto Albiol and Roberto Paredes for being the supervisors for this thesis. They have given me all the support I needed throughout these years.

To my parents and sister. They are the pillar of my education and the most important reason why I have been able to get here and for the person I have become.

To Aisha. Without her I'm nothing.

To my friends, specially David and Javi. Thanks for the support and the great times at the lab.

To Antonio Albiol and Mauricio Villegas. This work is also part of them.

To my cousin Carmina. For helping me with several tips for improving my writing.

To the Generalitat Valenciana - Consellería d'Educaciò for granting me an FPI scholarship, and to the Universitat Politècnica de València for being the host for my PhD.

Jordi Mansanet Sandín
Valencia, November 26th 2015

Contents

Abstract / Resumen / Resum iii

Acknowledgments vii

Contents ix

List of Figures xiii

List of Tables xv

Notation xvii

Abbreviations and Acronyms xix

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Overview of Contributions . . . 4
- 1.3 Thesis structure . . . 5

2 Overview of Deep Learning Methods 7
- 2.1 Historical Context . . . 7
- 2.2 Supervised Networks . . . 8
  - 2.2.1 Deep Neural Networks . . . 9
  - 2.2.2 Different types of units . . . 10
  - 2.2.3 The Backpropagation Algorithm . . . 11
  - 2.2.4 Deep Convolutional Neural Networks . . . 14
- 2.3 Unsupervised models . . . 15
  - 2.3.1 Restricted Boltzmann Machines . . . 16
    - 2.3.1.1 Introduction . . . 16
    - 2.3.1.2 Contrastive Divergence algorithm . . . 18
    - 2.3.1.3 Different type of units . . . 19
  - 2.3.2 Deep Belief Networks . . . 21
  - 2.3.3 Other unsupervised models . . . 23

3 Regularization Methods for RBMs **25**
3.1 Introduction . . . 25
3.2 Motivation and Contributions . . . 26
3.3 State of the Art . . . 28
3.4 Mask Selective Regularization for RBMs . . . 29
3.4.1 Introduction . . . 29
3.4.2 A Loss function combining L_2 and L_1 . . . 30
3.4.3 Mutual information and Correlation coefficient . . . 31
3.4.4 The binary regularization mask . . . 32
3.4.5 Topology selection and convergence . . . 35
3.4.6 The MSR Algorithm . . . 37
3.5 Experiments . . . 39
3.5.1 General protocol . . . 39
3.5.2 Experiments with MNIST . . . 40
3.5.3 Experiments with USPS . . . 44
3.5.4 Experiments with 20-Newsgroups . . . 46
3.5.5 Experiments with CIFAR-10 . . . 47
3.6 Conclusions . . . 48

4 Local Deep Neural Networks **51**
4.1 Introduction . . . 51
4.2 Motivation and Contributions . . . 52
4.3 State of the Art . . . 53
4.4 Local Deep Neural Networks . . . 55
4.4.1 Introduction . . . 55
4.4.2 Formal framework for local-based classification . . . 55
4.4.3 A local class-posterior estimator using a DNN . . . 57
4.4.4 Feature selection and extraction . . . 58
4.4.5 Location information and reliability weight . . . 60
4.5 Experiments . . . 60
4.5.1 General protocol . . . 61
4.5.2 Experiments with CIFAR-10 . . . 62
4.5.3 Experiments with MNIST . . . 66
4.6 Conclusions . . . 71

5 Application to Gender Recognition **73**
5.1 Introduction . . . 73
5.2 State of the Art . . . 75
5.3 Experiments . . . 76
5.3.1 General protocol . . . 76
5.3.2 Results with DNN . . . 77
5.3.3 Results with DCNN . . . 79
5.3.4 Results with Local-DNN . . . 81
5.3.5 Comparison of the results . . . 85
5.4 Conclusions . . . 86

6 General Conclusions **89**
6.1 Conclusions on Regularization Methods for RBMs 90
6.2 Conclusions on the Local-DNN model 90
6.3 Directions for Future Research 91
6.4 Dissemination 92

A Public Databases and Evaluation Protocols **95**
A.1 MNIST Database 95
A.2 USPS Database 95
A.3 20 Newsgroup Database 97
A.4 CIFAR-10 Database 97
A.5 Labelled Faces in the Wild Database 98
A.6 Groups/Gallagher Database 99

Bibliography **101**

List of Figures

1.1 Interpreting an image by extracting several intermediate representations. 3

2.1 Graphical representation of DNN model with two hidden layers. . . . 9
2.2 The Backpropagation algorithm for a neural network with two hidden layers. Note that bias terms have been ommited for simplicity. 13
2.3 Architecture of LeNet-5, a convolutional neural network for handwritten digits recognition. 15
2.4 Graphical representation of the RBM model. The grey and the white units correspond to the hidden and the visible layers, respectively. . . 17
2.5 Representation of the Markov chain used by the Gibbs sampling in the CD algorithm. 19
2.6 The sum of infinite copies of a sigmoid function with different bias can be approximated with the closed form function $\log(1 + e^x)$ denoted by the black line. The blue line represents the transfer function of a rectified linear unit. 21
2.7 Several RBMs are trained independently and stacked to form a DBN. This DBN also serves as a pre-training to a discriminative DNN. . . . 22

3.1 Graphical representation of the process to precompute the binary masks of the $\mathcal{M}$ set for the MNIST dataset. 34
3.2 Binary masks for $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ for the MNIST dataset. The red dots indicate the visible units related to each binary mask. 35
3.3 Some examples of binary regularization masks for the MNIST dataset. 35
3.4 Learned features for the MNIST dataset along with their corresponding binary masks overlaid in red color. 36
3.5 Graphical representation of the MSR method within the CD algorithm. 38
3.6 Clean and noise images 41
3.7 Histogram of weights for different regularization schemes in the first RBM for the MNIST dataset. 45
3.8 Comparison of a random selection of features learned by the RBM in the first layer. 46

4.1 Graphical depiction of the Local-DNN model. Several patches are extracted from the input image and they are fed into a DNN which learns a probability distribution over the labels in the output layer. The final label of the image is assigned using a fusion method that takes into account all the patches' contributions. 58
4.2 Feature extraction process using a sampling grid. 59
4.3 Feature extraction method that looks for high information content areas. 59
4.4 Probability of accuracy of the DNN at a patch level depending on the position where the patch was extracted for the CIFAR-10 dataset. Light and dark colors denote high and low probability, respectively. Note that the lowest probability is around 0.40 and the highest probability is around 0.59. 65
4.5 Probability of accuracy of the DNN at a patch level depending on the position where the patch was extracted for the MNIST dataset. Light and dark colors denote high and low probability, respectively. 69

5.1 Some feature detectors learned by the RBM model. 78
5.2 Graphical representation of a DCNN model. The parameters above denote the configuration used in the experiments of the gender recognition. 80
5.3 The 32 filters learned by the DCNN model in the first layer. 80
5.4 Probability of accuracy at patch level according to with the position where the patch was extracted. Light and dark colors denote high and low probability, respectively. 83

A.1 Some examples of handwritten digits of the MNIST dataset. 96
A.2 Some examples of handwritten digits of the USPS dataset. 96
A.3 Some examples of the CIFAR-10 dataset images. 98
A.4 Some examples of the LFW dataset images. 99
A.5 Some examples of the Gallagher dataset images. 99

List of Tables

3.1 Error rate (%) on the MNIST test set for both clean and noisy images using different regularization schemes and varying the number of labeled samples used in the fine-tuning. . . . 42
3.2 Effect of applying MSR to different layers of the network. Error rate (%) on the MNIST test set for both clean and noisy images. . . . 43
3.3 Error rate (%) on the USPS test set for both clean and noisy images using different regularization schemes. . . . 44
3.4 Error rate (%) on the 20-Newsgroups test set for both clean and noisy data. . . . 47
3.5 Error rate (%) on the CIFAR-10 test set. . . . 48

4.1 Number of images and approximate number of patches extracted from each subset in the CIFAR-10 dataset depending on the patch size. Note that $1K$ = one thousand and $1M$ = one million. . . . 62
4.2 Classification accuracy at image and patch level on the test set for the CIFAR-10 dataset, for different configurations of the Local-DNN model. 63
4.3 Classification accuracy on the test set of the CIFAR-10 dataset for different methods. . . . 66
4.4 Number of images and number of patches extracted from each subset in the MNIST dataset. . . . 67
4.5 Classification error rate at image and patch level on the test set for the MNIST dataset, for different configurations of the Local-DNN model. 68
4.6 Error rate on the test set of the MNIST dataset for different methods. 70

5.1 Accuracy on the test set for a one layer DNN in the LFW database. . 78
5.2 Accuracy on the test set for the DNN model with two and three layers in the LFW database. . . . 79
5.3 Accuracy on the test set for the DCNN model. . . . 81
5.4 Accuracy at patch level on the test set for the Local-DNN model varying the number of hidden layers. . . . 82
5.5 Accuracy on the test set for our Local-DNN model varying the number of hidden layers. . . . 83
5.6 Cross-database accuracy at image level using the Local-DNN model. . 84

5.7 Best accuracy on the test set for DNN, DCNN and Local-DNN models, and other state-of-the-art results. 85

A.1 The 20 classes of the 20 Newsgroup dataset, partitioned according to the subject matter. 97

Notation

In this thesis, the following notation has been used. Scalars are denoted in roman italics, generally using lowercase letters if they are variables (x, p, β) or in uppercase if they are constants (N, D, C). Also in roman italics, vectors are denoted in lowercase boldface ($\boldsymbol{x}$, $\boldsymbol{p}$, $\boldsymbol{\mu}$) and matrices in uppercase boldface ($\boldsymbol{X}$, $\boldsymbol{P}$, $\boldsymbol{B}$). Sets are either uppercase calligraphic ($\mathcal{X}$) or blackboard face for the special number sets ($\mathbb{R}$).

The following table serves as a reference to the common symbols, mathematical operations and functions used throughout the thesis.

Symbol	**Description**
$\hat{}$ (hat)	Complementary operator of a binary value.
σ_x	Standard deviation of the random variable x.
μ_x	Expected value of the random variable x.
Δx	Increment on the variable x.
$\langle x \rangle_y$	Expectation of the random variable x under the distribution y.
$\boldsymbol{A} \circ \boldsymbol{B}$	Hadamard or element-wise product between the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.
$\boldsymbol{A}^\mathsf{T}$	Transpose of the matrix $\boldsymbol{A}$.
$\boldsymbol{A}^{-1}$	Inverse of the square matrix $\boldsymbol{A}$.
$\hat{\boldsymbol{A}}$	Element-wise complementary of the binary matrix $\boldsymbol{A}$.
$a \in \mathcal{B}$	a is an element of the set $\mathcal{B}$.
$cov(x, y)$	The covariance between the random variables x and y.
$corr(x, y)$	The correlation coefficient between the random variables x and y.
$\|a\|$	Absolute value of the scalar a.
$\|\|\boldsymbol{a}\|\|_p$	The p-norm of the vector $\boldsymbol{a}$.

$\mathrm{sgn}(z) = \begin{cases} -1 & \text{if } z < 0 \\ 0 & \text{if } z = 0 \\ 1 & \text{if } z > 0 \end{cases}$	The signum function.
$\max(z) = \begin{cases} 0 & \text{if } z \leq 0 \\ z & \text{if } z > 0 \end{cases}$	The max rectifier function.
$\mathrm{step}(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases}$	The Heaviside or step function.
$\sigma(z) = \frac{1}{1+\exp(-z)}$	The sigmoid function.
$\delta(a, b)$	The *Kronecker* delta function.; $\delta(a, b) = 1$ if $a = b$; zero otherwise.
$\underset{c}{argmax}\ p\,(c\|\mathbf{x})$	The c argument which maximizes the value of $p\,(c\|\mathbf{x})$.

Abbreviations and Acronyms

AE	Auto-Encoder
ANN	Artificial Neural Network
CD	Contrastive Divergence
CDBN	Convolutional Deep Belief Network
CH	Color Histograms
CIFAR	Canadian Institute for Advanced Research
DAE	Denoising Auto-Encoders
DCNN	Deep Convolutional Neural Network
DL	Deep Learning
DBN	Deep Belief Network
DBM	Deep Boltzmann Machine
DNN	Deep Neural Network
EN	Elastic Net
FIPA	Facial Image Processing Group
GPU	Graphics Processor Unit
GRBM	Gaussian Restricted Boltzmann Machine
HOG	Histogram of Oriented Gradients
LBP	Local Binary Patterns
LDA	Linear Discriminant Analysis
LFW	Labeled Faces in the Wild
Local-DNN	Local Deep Neural Network
MI	Mutual Information
ML	Machine Learning
MNIST	Mixed National Institute of Standards and Technology
MSR	Mask Selective Regularization
NN	Neural Network
NReLU	Noisy Rectified Linear Unit
PCA	Principal Component Analysis
PCD	Persistent Contrastive Divergence
RBM	Restricted Boltzmann Machine
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Decent
SIFT	Scale-Invariant Feature Transform
SVM	Support Vector Machine

USPS	United States Postal Service
i.e.	*id est* (that is)
Acc.	Accuracy

Chapter 1

Introduction

The goal of this thesis is to present various contributions to the Machine Learning and computer science areas. All of these contributions emphasize on learning from data with the objective of extracting useful information. Despite the fact that *learning* is a very general concept, it can be defined as the task that aims to find a function that maps an input (e.g., a digital face image or a speech signal) to an output (e.g., the gender from the face image or the identity of the person who is talking). More specifically, this thesis is focused on addressing these kind of tasks by automatically learning several non-linear transformations of the data which are structured in layers. Since the number of layers can be high, these techniques are known in the literature as *Deep Learning* (DL).

The first part of the chapter strives to present the motivation behind this thesis and how DL methods can be used to solve challenging problems in diverse areas such as computer vision, speech recognition and natural language processing. Next, the second part of the chapter explains the main contributions presented in greater detail. Finally, the structure of the contents followed along the thesis is presented.

1.1 Motivation

As previously stated, one of the fields within Machine Learning (ML) focuses on the design of algorithms that aim to learn a function (or a mapping) from input data to an output value. This learning process involves discovering unknown probability distributions from data samples, and capturing statistical dependencies between the random variables that are used to represent the input data. However, this process might turn out to be difficult in some cases due to a number of factors.

First and foremost, the biggest challenge is the complexity and non-linear character that the mapping function may have due to the many variation factors that can appear in real world problems.

In this kind of problems, an enormous amount of data is usually required to ensure that there are enough samples to capture all of these factors of variation. Even with large amounts of data, some algorithms do not scale well due to the inherent limitation of the algorithm itself, or because of computational resources.

Another recurrent problem is that the learning complexity grows exponentially with the number of the input variables. 50 years ago, Richard E. Bellman called this phenomenon the *curse of dimensionality* [Bellman, 1957]. Some problems benefit from using *dimensionality reduction* techniques which reduce as much as possible the dimensionality of the data while keeping the significant information, even though this goal is sometimes hard to achieve.

The problems described above might be mitigated to some extent by using a different representation of the data. As suggested by [Bengio et al., 2013], the performance of ML methods is heavily dependent on the choice of the data representation on which they are applied. In order to make the machines to truly understand the world around us, it is necessary to identify and disentangle the underlying explanatory factors of variation hidden in the input data. This process aims at creating a more suitable representation of the data that eases the building of classifiers or other predictors. Actually, if we were able to know the best representation for each specific problem, the learning process would become quite a lot easier. For this reason, a considerable effort has been devoted in the last decades to design *human-engineered feature extraction algorithms* that aim to capture the useful information of the data in each specific task. This idea usually simplifies the mapping to be learnt and can lead to much better performance. However, the process of engineering features for each new application is arduous and it does not generalize well to other tasks.

Given the importance of using better representations of the data in the ML area, it would be desirable to be able to discover such representations automatically, hence new applications could be developed faster using a common general-purpose learning procedure. This is actually one of the goals behind the *Deep Learning* (DL) framework. Without going into the subject in depth, DL algorithms use Artificial Neural Networks (ANNs) to extract multiple levels of representations which disentangle the factors of variation in the data. These better representations correspond to more abstract or disambiguated concepts of the data that facilitate the process of learning. Despite the fact that ANNs have been already used for decades with less impact, they have seen a renaissance under the term *Deep Learning* partly thanks to the increase of the processing power and cheaper means of gathering more data. On the other hand, several advances and innovations have contributed to make DL very popular amongst researchers and industry, and very good results have been achieved in many tasks and domains. In this spirit, the motivation behind this thesis is to contribute to this process by presenting a work focused on several further tasks.

One of the keys of the DL framework is the use of deep architectures. Most ML models, such as Decision Trees, Support Vector Machines (SVM) or Naive Bayes, use shallow architectures that might be a shortcoming when dealing with real world problems. In contrast, there are several motivations towards using deep structures instead of shallow ones according to [Bengio and Courville, 2013]:

- *Brain inspiration*: a certain progress in neuroscience has discovered that the neocortex (an area of the brain associated with many cognitive abilities) does not pre-process sensory signals by itself, but rather propagates them through a complex hierarchy of levels [Lee et al., 1998].

- *Computational complexity*: some functions that can be represented compactly with a deep architecture would require an exponential number of components if they were represented with a shallow one [Bengio, 2009]. Obviously, the depth needed will depend on the task and the type of computation.
- *Cognitive arguments and engineering arguments*: humans organize ideas and concepts in a modular way and at multiple levels very often. Concepts at one level of abstraction are defined in terms of lower-level concepts. Also, most of the problems solved by engineers are tackled by typically constructing a chain or a graph of processing modules, where lower levels are the input of the upper ones.

After this brief motivation about the main features of the DL framework, it is interesting to resume these concepts with a simple example provided by Yoshua Bengio [Bengio, 2009]. The example regards the task of interpreting an input image such as one in Fig. 1.1. This image depicts a raw input made of many observed variables or

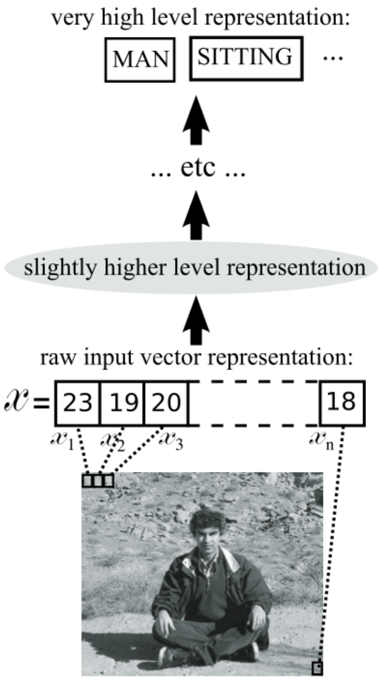

very high level representation:
MAN
SITTING
...
... etc ...
slightly higher level representation
raw input vector representation:
x =
23
19
20
18
x1
x2
x3
xn

Figure 1.1. Interpreting an image by extracting several intermediate representations.
Source: *Learning Deep Architectures for AI*, Yoshua Bengio (2009)

factors of variation such as the shadow, the background, the position of the man, etc.

These variations are related by unknown intricate statistical relationships which, unfortunately, cannot be taught to machines because we do not have enough formalized prior knowledge about the world. The categories *man* or *sitting* are an abstraction that may correspond to a very large set of possible images which can be very different between each other. DL aims to solve this problem by discovering automatically several lower-level and intermediate-level concepts or abstractions which would be useful to construct a *man sitting-detector*.

1.2 Overview of Contributions

The DL framework involves many types of algorithms and applications, and also the boundaries between what DL is and is not remain partly unclear. Most of the researchers in the DL community are specialized in specific topics that apply to their areas of interest. Like them, the major focus of this thesis is to share several contributions about different key factors that can play a role in the performance of several DL methods.

First of all, the first part of this thesis has been focused on the *regularization* process applied to the Restricted Boltzmann Machine (RBM) model. Regularization is a key concept not only in DL, but also in the Machine Learning area in general that prevents the models to overfit the training data, among other advantages. As it will be discussed later, one of the contributions of DL is the use of some prior knowledge that facilitates the training of deep architectures. This process is called *pre-traininig*, and it is usually done with unsupervised models like the RBM. At this point, these models have a huge number of parameters, so they can benefit from using regularization techniques. Our main contribution is to come up with a new regularization scheme called Mask Selective Regularization (MSR). This procedure is based on the idea of restricting the learning to specific parts of the data, offering several advantages compared to other traditional regularization methods. These assumptions have been validated empirically with several experiments in diverse application domains.

Following with the success obtained by using the idea of guiding the learning to useful regions of the data, the second part of the thesis somehow aims to use a similar idea in discriminative models. We present a new discriminative model called Local Deep Neural Network (Local-DNN) based on two key concepts: local features and deep architectures. In the case of images, one of the common problems is that, sometimes, it is difficult to directly learn from the entire image using neural networks. In contrast, our Local-DNN proposes to learn from small image regions called *patches*. This patch-based learning approach enhances the robustness of the network by using a probabilistic framework at the output that takes into account all the small contributions of each local feature. To compare the performance of this method, we have carried out several experiments in two well-known image datasets.

The last contribution of this thesis summarizes all the results obtained by an extensive experimental study using different DL models in the gender recognition task using face images. In these experiments we have also evaluated our Local-DNN model, which improves both the results obtained with other DL methods and obtains state-of-the-art results in the datasets evaluated.

1.3 Thesis structure

The remaining content of the thesis is organized as follows:

Chapter 2: this chapter reviews the most important models and elements included in the DL framework, which should be kept in mind throughout this thesis.

Chapter 3: this chapter introduces a new regularization method for RBMs called Mask Selective Regularization, and shows the experiments carried out.

Chapter 4: this chapter presents the novel Local Deep Neural Network model and draws the results obtained in the experiments performed.

Chapter 5: This chapter summarizes all the experiments performed in the gender recognition problem using DL methods.

Chapter 6: this chapter resumes and draws some conclusions about all the contributions of this thesis, and it also suggests some directions for future research following these lines.

Chapter 2

Overview of Deep Learning Methods

In this chapter, we put into perspective several elements of Deep Learning (DL) algorithms that are related to this thesis. However, to obtain a broader view of the current trends on this topic, the reader is encouraged to read an excellent review presented recently by three of the most important researchers in this area [LeCun et al., 2015].

First of all, the historical context around DL is described briefly for a better understanding of its development until these days. After that, the following section is focused on two important supervised models: the Deep Neural Network and the Deep Convolutional Neural Network. Finally, the last section mainly describes one of the most important unsupervised models in this thesis (the Restricted Boltzmann Machine (RBM)), and the Deep Belief Network model which is created by stacking several RBMs.

2.1 Historical Context

Deep Learning (DL) models are based on Artificial Neural Networks (ANNs). Actually, some researchers call DL *the new generation of neural networks*. Historically, ANNs started in 1943, when the neurophysiologist Warren McCulloch and the mathematician Walter Pitts modeled a simple neural network using electrical circuits to explain how the neurons might work in the brain. The first great results arrived in the late 50s and early 60s when the scientist Frank Rosenblatt created the *perceptron*, a linear model that combines a set of weights (parameters) with an input vector to perform a binary classifier. Using this model, the first ANN applied to a real world problem was proposed by Bernard Widrow and Macian Hoff to eliminate echoes on a phone line. At the same time, one of the most important moments took place when several researchers suggested the Backpropagation (BP) algorithm to automatically learn the parameters of these networks [Rumelhart et al., 1988; Werbos, 1974]. After a big apogee between the mid 80s and 90s, the ML community steered away from ANNs and started focusing on other methods like Support Vector Machines (SVMs), Linear models, Maximum Entropy models, etc. The main problem with ANNs models was in the training stage. With the computational power available it was very difficult to take advantage of these networks, and the learning process was too slow. Also, this learning process was based on optimizing a non-convex error function, which can be an issue due to the presence of several local minimum values. Furthermore, the training sets in those days were usually small, and the networks were not able to generalize well to new samples (the *overfiting* problem). For these reasons, amongst many others, only a few groups continued working with ANNs with limited scale and impact [Bengio and Bengio, 2000; LeCun et al., 1998; Rumelhart et al., 1988].

However, nowadays ANNs are back in fashion under the term *Deep Learning*. One of the milestones in this story occurred in 2006 when Geoffrey Hinton's lab was able to train efficiently a deep network able to reconstruct high-dimensional input vectors from a small central layer [Hinton et al., 2006; Hinton and Salakhutdinov, 2006]. The main goal of this work was to demonstrate empirically that initializing the weights of this network using unsupervised models, often produces much better results than the same network initialized with random weights. This discovery was a huge breakthrough in the research community, and, from that moment on, DL and neural networks have been receiving more and more attention progressively until these days. Another important stage in the DL history occurred in the ImageNet competition of image understanding in 2012. A Deep Convolutional Neural Network model was applied to a dataset of about a million images, and the results obtained almost halved the error rates of the best competitors [Krizhevsky et al., 2012]. This success showed the power of these models dealing with images when they can be trained efficiently using graphics processing units (GPUs) with tons of labeled data. Actually, from that moment on, DCNNs have become the dominant approach for almost all the recognition and detection computer vision tasks [Sermanet et al., 2014; Taigman et al., 2014; Tompson et al., 2014; Zeiler and Fergus, 2013]. Also, not only in the academic domain but also some of the most important companies in the world, like Google, Facebook and Baidu, have led big advances in image and speech recognition using DL technologies.

2.2 Supervised Networks

In this section we address the supervised learning, which is the most common type of learning not only in Deep Learning but also in the Machine Learning in general. In this type of learning, the desired output of the training data is known in advance, and it is supplied to the model during training. Therefore, each training sample is represented as a *pair* consisting of an input vector represented by $\mathbf{x}$, and a desired output value represented by y. The algorithm should infer a function which can be used for mapping samples with unknown outputs. Within a probabilistic framework, this kind of models are also known as *discriminative models* because they model the conditional probability distribution $P\left(y|\mathbf{x}\right)$ which can be used for predicting y from $\mathbf{x}$.

2.2.1 Deep Neural Networks

An Artificial Neural Network (ANN) is a generic term to encompass any structure of interconnected neurons which sends information between each other. However, in this thesis we will refer to Deep Neural Networks (DNN) as a discriminative network with one input layer, one or more hidden layers, and one output layer. Note that a network with just one hidden layer cannot actually be considered *deep*. However, in this thesis we have included this type of network under the term DNN for convenience. Actually, the essence of the network architecture is the same despite the number of hidden layers, and it allows us to present the results changing this parameter in a unified manner. Fig. 2.1 shows a graphical depiction of a DNN with two hidden layers. As can be seen in the figure, this kind of network is *fully-connected* and *feed-*

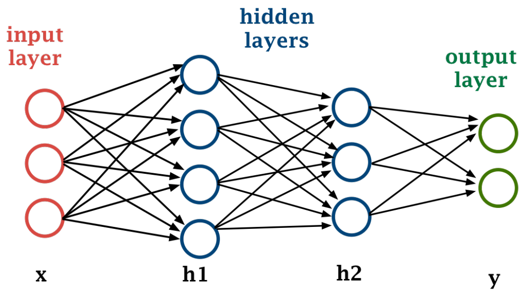


Figure 2.1. Graphical representation of DNN model with two hidden layers.

forward. The former term means that each neuron in one layer is connected to all the neurons in the next layer. The later term means that there are not cycles, so that the connections do not form feedback loops. The input layer, denoted by $\mathbf{x}$, represents the input data vector, so it has as many neurons as dimensions of the input space. For instance, in the case that the input sample is an image, all the pixels are vectorized and each input neuron represents one of those pixels. This raw data is transformed into new representations using the hidden layers, denoted by $\mathbf{h1}$ and $\mathbf{h2}$. The weights between the layers and the non-linearities introduced by the neurons work as a feature extractors. They should produce representations that are relevant to the aspects of the data that are important for discrimination. Both the number of hidden layers and the number of units in these layers are not predefined, and they should be estimated empirically depending on the problem at a hand. Finally, the output layer, denoted by $\mathbf{y}$, represents the labels of the samples. The number of output units corresponds with the number of different classes in the discriminative problem. For instance, a network to classify handwritten digits (0, 1, 2, ..., 9) must contain an output layer with 10 neurons.

More formally, if the number of layers of the network is denoted by L (being $L = 0$ the input layer), the mapping function of the transformations performed by the network can be defined as

$$f(\mathbf{x}) = f_L\left(f_{L-1}\left(\dots f_1(\mathbf{x})\right)\right) \ . \tag{2.1}$$

Each of these transformations depend on the input vector to the layer and the parameters of the network represented by $\theta = (\mathbf{W}, \mathbf{b})$, where $\mathbf{W}$ is a matrix that represents all the weighted connections between two layers and $\mathbf{b}$ is a vector that represents the bias term. Therefore, the transformation in the layer l is given by

$$f_l(\mathbf{v}) = g(\mathbf{W}\mathbf{v} + \mathbf{b}) \ , \tag{2.2}$$

where $\mathbf{v}$ is the input vector to the layer and $g(\cdot)$ is called the *activation function*. This activation function is defined accordingly to the type of unit employed in the network. The most popular types of units are explained in the next section.

2.2.2 Different types of units

One of the most interesting properties of neural networks is resumed in the *universal approximation theorem* [Cybenko, 1989; Hornik et al., 1989]. This theorem states that a simple feed-forward network with a single hidden layer can approximate any continuous function given the appropriate parameters. This powerful characteristic is partly due to each non-linearity introduced by the *activation function* related to the type of unit (neuron) used in the network. The activation function is also known as the *transfer function* because it computes the output value of the unit given its input, which is a weighted sum of the outputs from the previous layer.

The first artificial neuron proposed in the context of neural networks was the *perceptron* (see Section 2.1). The activation function of this type of neuron is given by the Heaviside step function:

$$g(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z > 0 \end{cases} \ , \tag{2.3}$$

where z is the input value of the neuron. However, one of the problems of the perceptron is that a small change in a weight (or bias) can sometimes cause the output of the unit to completely flip, say from 0 to 1. That makes it difficult to see how to gradually modify the weights and biases during learning so that the network gets closer to the desired behaviour.

It is possible to overcome this problem by using the *sigmoid* neuron, which is the most conventional non-linearity employed in neural networks. Its transfer function is given by

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

This type of neuron facilitates the process of learning, hence small changes in the parameters of the model (weights and bias) cause only a small change in the output of the neuron [Nielsen, 2015]. Note that this type of function is also called the *logictic function* and it is usually represented by the symbol σ.

One of the problems of using sigmoid units in neural networks occurs when the input value of the neuron is too small or too big (saturation) because the gradient becomes very small. This is known as the *vanishing gradient problem*, which implies that the learning process takes too much time [Bengio and Glorot, 2010; Hochreiter et al., 2001]. However, the use of other type of unit such as the Rectified Linear Unit

(ReLU) may alleviate this problem. The transfer function of this neuron is simply the rectifier given by

$$g(z) = max(z, 0) \ . \tag{2.5}$$

In the last years, several experiments have shown the advantages of this type of neurons. For instance, the ReLU unit learns much faster than the standard sigmoid in networks with many layers during the training process [Glorot et al., 2010; Goodfellow et al., 2013; Zeiler et al., 2013]. Actually, the ReLU unit is currently the most popular non-linearity employed in neural networks.

The type of units explained above are usually employed in the hidden layers. These non-linearities allow the network to change the representation space of the data to facilitate the separation of the samples amongst the different classes. However, the last layer of the network must encode the labels of the samples. This is done using the *softmax function* which assigns a value to the *j-th* output unit given by

$$g(z_j) = \frac{e^{z_j}}{\sum\limits_{k=1}^{K} e^{z_k}} \ . \tag{2.6}$$

where z_k is the input value of the *k-th* unit and K is the total number of output units. According to the function, the output from the softmax layer is a set of positive numbers which total 1. In other words, this layer maps the output of the previous layer to a conditional probability distribution of the possible labels. Given that, the predicted label by the network corresponds to the output neuron with the highest probability value.

Although the types of units summarized in this section are the most popular in neural networks, many others can also be used. For more information, an extensive study of different units in the exponential family can be read in [Welling et al., 2005].

2.2.3 The Backpropagation Algorithm

During the training process of supervised neural networks, the model is fed with a data sample and produces an output in the form of a vector of probabilities also known as *scores*, one for each class. The goal of this process is that the correct class of the sample should have the highest score among all the classes. However, this is unlikely to happen without modifying the initial parameters of the network.

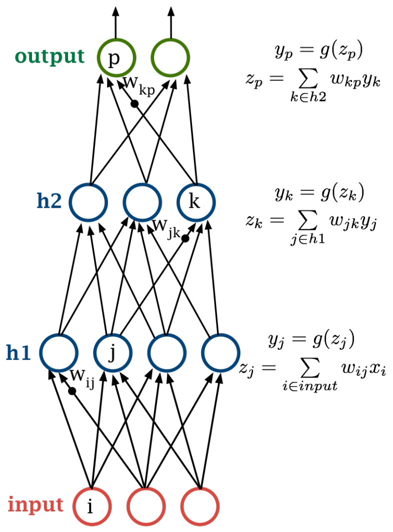
Several groups proposed the Backpropagation (BP) algorithm during the 1970s and 1980s [Rumelhart et al., 1988; Werbos, 1974] to automatically modify the parameters of the model and make the network to learn the desired category of the samples. The intuitive idea behind this method is to minimize an objective function that measures the error between the current output scores and the true vector of scores (the label of the sample). The algorithm should modify the internal parameters of the network (weights and bias) to reduce this error. To learn how these weights must be modified, the learning algorithm computes a gradient vector that indicates the amount of increased or decreased error obtained when the weight value is slightly modified. This process can be seen as a search of the minimum value of the error function in a high-dimensional space defined by the weights values. The computed gradient indicates the direction of the steepest descent towards that state of the weights which gives the minimum error.

When this algorithm is applied to a neural network, this process is performed through all the layers of the network. The first part of the algorithm is called the *forward* pass. This part aims at computing all the units activations by propagating the input data to the upper layers. The input z of each unit is computed as the weighted sum of the outputs of the units in the layer below. Then, one of the transfer functions defined in Section 2.2.2 is applied to this value to obtain the output of the unit, denoted by y. The equations of these process are depicted in Fig. 2.2a.

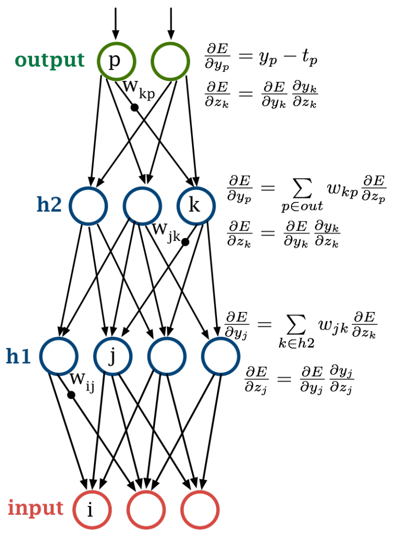
Once the output values of the last layer are computed, it is possible to calculate the error (E) between the predicted value (y) and the desired value (t). A common way to compute this discrepancy is using the squared error measure, so that $E = 0.5(t - y)^2$. This is the beginning of the second part of the algorithm, called the *backward* pass. Basically, this process consists of propagating the error through the network computing the error derivatives with respect to the weight parameters to know how the parameters must be updated. To that end, the *chain rule* is used, which is a formula for computing the derivative of a composed function. An example of this process for a two-layer neural network is represented in Fig. 2.2b. Note that these equations are subject to change if other error function is employed, such as the *cross-entropy* cost function [Golik et al., 2013]. A detailed explanation of this algorithm with all the mathematical derivations can be found in [Nielsen, 2015].

The training of a neural network using the BP algorithm is performed by using a training set of data samples. A common procedure is to use a Stochastic Gradient Decent (SGD) method. This procedure computes the average gradient for a small set of examples, and the weights are adjusted accordingly. This process is repeated for many small sets of examples from the training set until the average error stops decreasing. Typically, the term *batch* is used to define these small sets of samples. Also, the term *epoch* is used as a measure of the number of times that all the training samples are used once to update the weights. In practice, it is common to train the model for several epochs, so that the network *sees* the entire training set many times.

As stated in Section 2.1, neural networks were mostly ignored between mid 90s and 2006 by the Machine Learning community. It was thought that the BP algorithm would get trapped in poor local minima due to the non-convex nature of the error function to be minimized. Also, it was difficult to propagate the error through the layers when the network is deep, which is known as the *vanishing gradient problem* [Hochreiter et al., 2001]. However, nowadays these problems have been minimized for several reasons. First of all, the use of ReLU units instead of the traditional sigmoids have allowed to train efficiently deeper networks. Also, the increase of computational resources have boosted the learning process, specially with the use of GPUs, so that huge training sets can be processed. Finally, the neural networks achieve their full potential with a huge number of data samples, which was more difficult to obtain in the past.


output
p
w_{kp}
$y_p = g(z_p)$
$z_p = \sum_{k \in h2} w_{kp} y_k$
h2
k
w_{jk}
$y_k = g(z_k)$
$z_k = \sum_{j \in h1} w_{jk} y_j$
h1
j
w_{ij}
$y_j = g(z_j)$
$z_j = \sum_{i \in input} w_{ij} x_i$
input
i

(a) The forward pass in the Back-propagation algorithm.


output
p
w_{kp}
$\frac{\partial E}{\partial y_p} = y_p - t_p$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$
h2
k
w_{jk}
$\frac{\partial E}{\partial y_p} = \sum_{p \in out} w_{kp} \frac{\partial E}{\partial z_p}$
$\frac{\partial E}{\partial z_k} = \frac{\partial E}{\partial y_k} \frac{\partial y_k}{\partial z_k}$
h1
j
w_{ij}
$\frac{\partial E}{\partial y_j} = \sum_{k \in h2} w_{jk} \frac{\partial E}{\partial z_k}$
$\frac{\partial E}{\partial z_j} = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial z_j}$
input
i

(b) The backguard pass in the Back-propagation algorithm.

Figure 2.2. The Backpropagation algorithm for a neural network with two hidden layers. Note that bias terms have been ommited for simplicity.
Source: *Deep Learning*, YannLeCun, Yoshua Bengio and Geoffrey Hinton (2015)

2.2.4 Deep Convolutional Neural Networks

Despite the great advances using DNNs, there are still some problems that are difficult to solve using this kind of networks. The layers in a DNN are *fully-connected*, which means that all the units between adjacent layers are connected between each other. This property could be a problem if the dimensionality of the input space was large, which usually happens in computer vision problems. As an example, to model an image of size 200×200 pixels with a hidden layer of 4,000 units, we need 1.6×10^8 parameters. Obviously, this network would be very difficult to train even with a shallow architecture with just one layer. Not to mention the inherent problem of overfitting due to the *plasticity* of the network to be able to approximate any complex function, as stated in the beginning of Section 2.2.2. Also, trying to model each pixel of the image with a dedicated connection might not work well in practice, specially with complex data obtained in unconstrained scenarios [Krizhevsky, 2009].

All of these problems have promoted extensively the use of other type of discriminative model called Deep Convolutional Neural Networks (DCNN) in computer vision problems. The convolutional neural network was first introduced a long time ago by [Fukushima, 1980; LeCun et al., 1998]. Despite this, its application has been mostly limited to problems related to character and handwriting recognition systems until these days [LeCun et al., 1998].

The architecture of this type of network is similar to other networks because they contain several hidden layers where each layer applies an affine transformation to the input data followed by a non-linearity. DCNNs leverage these three key ideas: *local connectivity*, *parameter sharing* and *pooling layers*. The first one, *local connectivity*, attempts to reduce the number of parameters by connecting each hidden unit only to a subregion of the input image. This process exploits the spatial stationarity assumed in the topology of the data (in this case, pixels), to greatly reduce the number of parameters. The second one, *parameter sharing*, exploits the idea of extracting feature detectors in several parts of the image that may be useful elsewhere. In other words, the feature detectors obtained are *equivariant* because the same feature can detect patterns in different parts of the image. This also reduces the number of free parameters and it rises the computation efficiency. Finally, the use of *pooling/subsampling* layers merges the activations of many similar feature detectors into a single response. For instance, *max-pooling* is a typical way of combining these responses in which the non-linear *max* function outputs the maximum value across several sub-regions from the given representation. These pooling layers help the network to be robust to small translations of the input, aside from reducing the number of hidden units. Typically, the first layers of a DCNN are pairs of convolutional and pooling layers, which forms a powerful feature extractor block. At the end of the network, one or more fully-connected layers are commonly added . These fully-connected layers capture non-linear relations between the higher-level representations of the image. Fig. 2.3 represents the convolutional network LeNet-5 implemented by Yann Lecun in 1998, which is very well known for its success in the digit recognition task. Note that this kind of network can also be trained using the Backpropagation algorithm.

Despite the power of this DCNN model, it was forsaken by the computer vision and machine learning communities for many years. In 2012, this fact changed due to

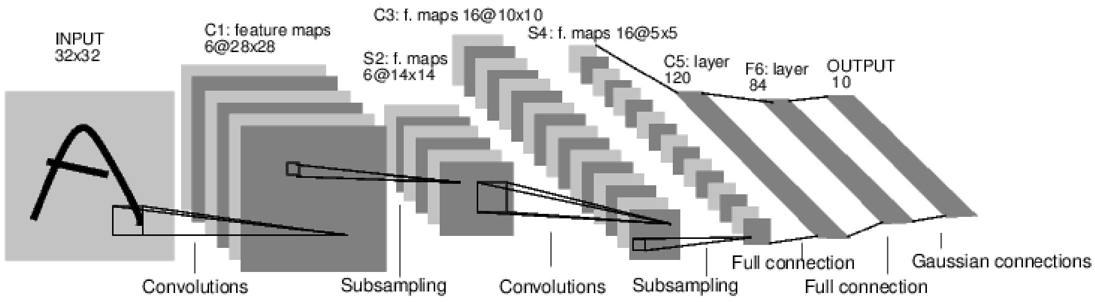


Figure 2.3. Architecture of LeNet-5, a convolutional neural network for handwritten digits recognition.
Source: *Gradient-Based Learning Applied to Document Recognition*, Yann Lecun (1998)

the spectacular results obtained in the ImageNet competition using a convolutional network [Krizhevsky et al., 2012]. The network employed was able to recognize objects in a dataset of about one million images among 1000 different classes, almost halving the error rates of the best methods evaluated. In 2013, [Zeiler and Fergus, 2013] later improved these results using a larger DCNN, and most of the approaches presented in the competition made use of these convolutional networks. This success is mainly due to the increase of computational resources, specially the efficient use of GPUs, and the larger sets of labelled data available. As happened with the standard DNNs, the ReLU units are also widely used in this type of networks. Also, other improvements such as the *dropout* method have become a key factor of this success [Srivastava et al., 2014]. This method aims to mitigate the typical overfitting problem by randomly dropping units (along with their connections) during training. This prevents the units from co-adapting too much, which causes the network to not generalize well to new samples. These improvements have made the convolutional networks as the dominant approach for almost all the image recognition and detection tasks in these days [Krizhevsky et al., 2012; Russakovsky et al., 2015; Zeiler and Fergus, 2013]. It should be highlighted that in some tasks, the performance obtained is similar to that reported for humans, for instance in the face recognition problem [Taigman et al., 2014].

The success of this type of networks inspired us to develop the Local-DNN model presented in Chapter 4. Our model somehow follows a similar idea of learning from local regions in the image, but learning local DNN networks.

2.3 Unsupervised models

Unlike the supervised models presented in Section 2.2, this section is focused on other type of networks called *unsupervised models*. These models aim to discover the hidden structure of unlabeled data. From a probabilistic point of view, this kind of learning is related to the problem of *density estimation*, which deals with the estimation of the probability distribution of the input data.

One of the breakthroughs in the development of Deep Learning (DL) techniques was the use of *pre-training* to allow a more efficient training of deep networks [Hinton and Salakhutdinov, 2006]. The idea is that each block of a deep network can be pre-trained using an unsupervised model. Each block captures regularities from its input distribution without requiring labelled data. This process is done layer-wise, and the parameters learned in each block serve as a good initialization of the weights of that block in a deep neural network.

This idea can also be seen from a probabilistic point of view. Consider random input-output pairs (x, y) in a neural network. Learning a mapping function between both of them involves modeling an approximation of the probability distribution $P(y|x)$ by maximizing its likelihood. If the true $P(x)$ and $P(y|x)$ are related[1], learning $P(x)$ may facilitate the modeling of the real target distribution $P(y|x)$ [Bengio, 2009].

2.3.1 Restricted Boltzmann Machines

2.3.1.1 Introduction

Despite the fact that there are several unsupervised models based on different approaches, in this thesis we focus our attention on the Restricted Boltzmann Machine (RBM) model [Freund and Haussler, 1991; Hinton, 2002; Smolensky, 1986]. For a precise description of this model we can enumerate its four main characteristics. First, the RBM is an unsupervised model, so its training procedure does not involve any class information about the samples[2]. Second, the RBM is a probabilistic model, so it attempts to learn a probability distribution over its inputs. Third, the RBM is a generative model, which means that the model can be used to meaningfully generate samples once the probability manifold of the data has been learnt. Finally, the RBM is an energy-based model, which means that the model captures dependencies between variables by associating a scalar energy to each configuration of those variables.

Going into more detail, the RBM model can be defined as a *probabilistic graphical model* because it can be represented using a graph that expresses the conditional dependence structure between random variables. These random variables are represented by two layers of units connected by means of several weights. The first layer is called *visible* because represents the input data, and the second one is called *hidden* because it represents latent variables that increase the expressiveness of the model. A graphical representation of this model can be seen in Fig. 2.4. Note that there is no connection from hidden units to other hidden units, nor from visible units to other visible units, unlike the original Boltzmann Machine model [Hinton and Sejnowski, 1986].

The simplest RBM is one in which all the units are binary. In this case, every pair of visible ($\mathbf{v} \in \mathbb{R}^D$) and hidden ($\mathbf{h} \in \mathbb{R}^N$) states have an energy value given by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{2.7}$$

[1]For example, some digit images form well-separated clusters using clustering algorithms. So that, the decision surfaces can be guessed reasonably well even before seeing any label.

[2]Note that [Larochelle and Bengio, 2008] illustrates how RBMs can also be used for classification.

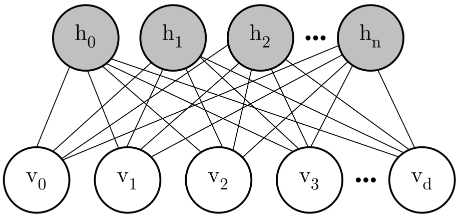


Figure 2.4. Graphical representation of the RBM model. The grey and the white units correspond to the hidden and the visible layers, respectively.

where v_i and h_j are the binary states of the visible unit i and the hidden unit j, w_{ij} is the weight of the connection between them, and a_i and b_j are the biases of those units. The relation between this energy value and the probability assigned to that pair of visible and hidden vectors is given by:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} e^{-E(\mathbf{v},\mathbf{h})} , \tag{2.8}$$

where the *partition function*, denoted by Z, is given by summing over all possible pairs of visible and hidden vectors:

$$Z = \sum_{\mathbf{v},\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} . \tag{2.9}$$

The probability that the network assigns to a visible vector $\mathbf{v}$ is given by summing over all possible hidden vectors:

$$p(\mathbf{v}) = \frac{1}{Z} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})} . \tag{2.10}$$

The objective of learning in the RBM model is to raise the probability that the network assigns to a training sample (lower its energy), and to lower the probability assigned to other samples (rise its energy). The log-likelihood of the training data is given by

$$\mathcal{L}(\theta, \mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p(\mathbf{x}) , \tag{2.11}$$

where θ are the parameters of the model (weights and biases) and $\mathbf{x} \in R^D$ is a training sample of the dataset $\mathcal{D}$. During the training process the parameters of the model are adjusted so that $\mathcal{L}(\theta, \mathcal{D})$ is maximized. This optimization problem is solved by estimating the gradient of this log-likelihood function with respect to the parameters. The derivative of the log-probability of a training sample with respect to a weight yields a very simple expression (see [Krizhevsky, 2009] for details on how this gradient expression is derived):

$$\frac{\partial \log p(\mathbf{x})}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} , \tag{2.12}$$

where the angle brackets are used to denote expectation under the distribution specified by the subscript that follows. In other words, $\langle v_i h_j \rangle_{data}$ is the frequency of the visible unit i and the hidden unit j being active together when the model is driven by samples of the training set, and $\langle v_i h_j \rangle_{model}$ is the corresponding frequency when the model is let free to generate likely samples (not driven by any data). This leads to a very simple learning rule to update the parameters of the model:

$$\Delta w_{ij} = \epsilon\big(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}\big) \;, \tag{2.13}$$

where ϵ is a learning rate. Note that this learning rule is applied to perform a stochastic gradient ascent process in the log probability of the training data. A simplified version of the same learning rule is used for the biases.

Due to the fact that there are no direct connections between the units in the hidden layer, it is very easy to get an unbiased sample of $\langle v_i h_j \rangle_{data}$. The process is composed of two steps. First, clamping a random training vector $\mathbf{v}$ in the visible layer. Then the binary state h_j of the hidden unit j is computed by sampling stochastically from a Bernoulli distribution with a probability given by

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i w_{ij} v_i) \;, \tag{2.14}$$

where $\sigma(x)$ is the transfer function for the sigmoid units employed in the standard binary RBM. Likewise, it is also very easy to get an unbiased sample of the state of a visible unit v_i, given the hidden vector $\mathbf{h}$ previously calculated:

$$p(v_i = 1|\mathbf{h}) = \sigma(a_i + \sum_j w_{ij} h_j) \;. \tag{2.15}$$

By performing these two steps, an unbiased sample of the first term of the expression is already computed 2.13.

However, it is much more difficult to get an unbiased sample of $\langle v_i h_j \rangle_{model}$ because that would require to know the true probability distribution of the model. We can approximate that distribution using a Markov Chain Monte Carlo algorithm based on the Gibbs sampling method [Geman and Geman, 1984]. This method starts at any random state of the visible units and updates alternatively the hidden and the visible states several times using the Eqs. 2.14 and 2.15. However, we must run these updates for a very long time until the model reaches the equilibrium and the statistics become unbiased. This problem is addressed by the Contrastive Divergence algorithm, which is described in the next section.

2.3.1.2 Contrastive Divergence algorithm

A much faster learning procedure called Contrastive Divergence (CD) was proposed by [Hinton, 2002]. The basic idea of this algorithm is that there is no need to have an accurate estimate of the gradient, as long as this estimation is in the right direction.

As explained before, the hardest part of training an RBM is approximating the second term of the gradient (see Eq. 2.12), i.e. $\langle v_i h_j \rangle_{model}$. To overcome this problem, the CD algorithm uses two tricks. On the one hand, the process is speeded up by starting the Gibbs sampling at the training data instead of starting at any random state. The reason is that, starting randomly and after some reasonable time, the distribution learned by the RBM will be close to the training data, so starting already at the training data indeed achieves a quick start. On the other hand, it is assumed that it is usually enough to estimate the direction of the gradient, even though the magnitude of that gradient might be inaccurate. Therefore, CD does not wait for the sampling process to converge, so that the gradient is estimated after only *k-steps* of the Gibbs sampling. The learning works well for small values of k despite the fact that that approximation is far from the true likelihood gradient [Hinton, 2002; Sutskever and Tieleman, 2010]. Actually, $k = 1$ is enough for most applications, what is known as the CD_1 algorithm. An extensive numerical comparison of training with CD_k versus exact log-likelihood gradient is presented in [Carreira-Perpinan and Hinton, 2005]. Fig. 2.5 shows a graphical representation of the CD algorithm. As we can see, the statistics in the activities of a visible and a hidden unit are measured after the first update of the hidden unit and again at the end of the chain.

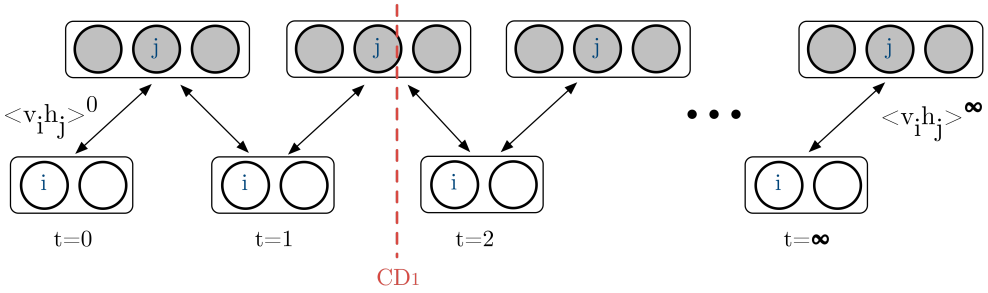


Figure 2.5. Representation of the Markov chain used by the Gibbs sampling in the CD algorithm.

A modified version of CD called Persistent Contrastive Divergence (PCD) was proposed by [Tieleman, 2008a]. This method does not initialize each alternating Gibbs Markov chain with a different training sample, but rather keeps track of the states of a number of persistent chains called *fantasy particles*.

2.3.1.3 Different type of units

RBMs were originally developed using binary visible and hidden units, and everything stated up to now is based on this assumption. However, other types of units have been recently proposed. An extensive study of different units in the exponential family is given in [Welling et al., 2005]. The following describes the two types of units employed in this thesis, besides the standard binary unit.

Gaussian units

Gaussian units are also known as *linear units* because their transfer function is given by the linear function $g(z) = z$. In this case, the sampling process is performed by adding a gaussian noise to the unit value during training. This type of unit is usually employed in the visible layer of the RBM when the data to be modeled has continuous values, such as pixels or the Mel-Cepstrum coefficients used to represent speech. When the RBM model uses gaussian units in the visible layer and binary units in the hidden layer, the energy function of this Gaussian-RBM (GRBM) model is given by

$$E_{GRBM}(\mathbf{v},\mathbf{h}) = \sum_{i\in vis} \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{j\in hid} b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} \ , \tag{2.16}$$

where v_i denotes the real-valued activity of the visible unit i, σ_i is the corresponding standard deviation of the Gaussian noise, and h_j is the binary state of the hidden unit j. Like in the standard RBM, the hidden units are mutually independent given the visible units and vice versa, so that the conditional distribution over the hidden units remains basically unchanged:

$$p(h_j = 1|\mathbf{v}) = \sigma(b_j + \sum_i w_{ij} \frac{v_i}{\sigma_i}) \ . \tag{2.17}$$

The only difference from Eq. 2.14 in the standard RBM is that v_i is scaled by the σ_i factor. However, the inverse conditional probability distribution is quite different and it is given by sampling from a Gaussian distribution:

$$p(v_i|\mathbf{h}) = \mathcal{N}(\mu_i, \sigma_i^2) \ , \tag{2.18}$$

where $\mu_i = a_i + \sigma_i^2 \sum_j w_{ij} h_j$. Therefore, the probability distribution for a single visible unit i given $\mathbf{h}$, is an independent normal distribution with mean μ_i and variance σ_i^2.

The GRBM can be trained with the CD algorithm as well using the equations 2.17 and 2.18 to perform the Gibbs sampling. Also, It is possible to learn the parameter of the variance of the noise for each visible unit (σ_i) using the CD_1 algorithm [Krizhevsky, 2009]. However, learning this parameter is generally difficult and takes long time. This is why some authors argue that, in most applications, it is much easier to first normalize each component of the data to have zero mean and unit variance and then use noise free reconstructions, with $\sigma_i = 1$ [Hinton, 2010; Hinton and Salakhutdinov, 2006; Nair and Hinton, 2010]. In this thesis, we have followed this approximation in the experiments using the GRBM model.

Noisy Rectified Linear Units

One of the possible disadvantages of the binary units in the visible and the hidden layers can be given by their poor representation power, since each unit captures just one bit of information. To allow each unit to encode an integer value, [Teh and Hinton, 2001] introduced the binomial units, which can be viewed as N identical copies of a binary unit that all share the same bias and weights. During the reconstruction of the data from the hidden activities, all the replicas have the same probability (p) of turning on. Therefore, the total number of "on" copies is modeled with a binomial distribution with mean Np and variance $Np(1-p)$. The problem is that as p approaches 1 the variance becomes small, which may not be desirable.

A small modification to binomial units makes them far more interesting to model real neurons and also more useful for practical applications. We can make infinite copies with the same weights and bias (b), but each bias is modified by summing a fixed offset of 0.5, i.e. $b - 0.5, b - 1.5, b - 2.5, \ldots$. The sum of the probabilities of the copies is close to having a closed form, as depicted in Fig. 2.6. This set of copies has no more parameters than an ordinary binary unit, but it is able to represent integer noisy values along the curve $\log(1 + e^x)$. A drawback of this approach is that the

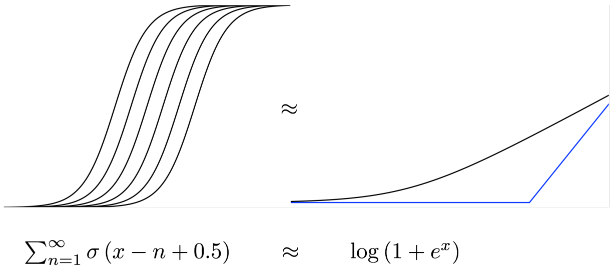


Figure 2.6. The sum of infinite copies of a sigmoid function with different bias can be approximated with the closed form function $\log(1 + e^x)$ denoted by the black line. The blue line represents the transfer function of a rectified linear unit.

logistic function needs to be used many times to sample an integer value correctly. It is possible, however, to use a fast approximation based on a rectified linear unit, whose transfer function is given by $max(0, x + \mathcal{N}(0, \sigma(x)))$ where $\mathcal{N}(0, V)$ is a Gaussian noise with zero-mean and a variance V that is added during the sampling process. This type of unit is called *Noisy Rectified Linear Unit* (NReLU), and its representation capacity is not constrained to be an integer value. One nice thing of using NReLU units is that the mathematics underlying the original RBM remain unchanged. [Nair and Hinton, 2010] showed that using this type of units in the hidden layer works much better than using binary units for recognizing objects and comparing face images, and they can deal with large intensity variations much more naturally than binary units.

2.3.2 Deep Belief Networks

The RBM model can be considered a shallow model because it has only one hidden layer. As stated in Chapter 1, Deep Learning's main contribution is based on using deep architectures to face more complex problems in a more efficient way. Unsupervised models also benefit from deep structures, and the Deep Belief Network (DBN) model is a good example of this fact. A DBN is a deep generative model composed of a visible layer and multiples hidden layers of latent variables. There are connections between the layers but not between units within each layer.

Before 2006, deep generative models based on neural networks were called Sigmoid Belief Networks. These models were usually hard to train because the optimization process involved the training of the entire network at a time using variational approximations [Dayan et al., 1995; Hinton et al., 1995]. However, G. Hinton and R. Salakhutdinov introduced a greedy layer-wise unsupervised learning algorithm that helped to optimize these deep networks [Bengio et al., 2007; Hinton and Salakhutdinov, 2006]. As already mentioned in this thesis, this process is called *pre-training*. The pre-training can be performed using several RBMs if the deep network aims to be a generative model[3]. The number of RBMs needed is related to the number of hidden layers in the deep network. Therefore, a DBN can be defined as a composition of simple RBMs, where each RBM's hidden layer is the visible layer for the upper one. The bottom RBM model can capture simple factors of variation from the raw data, which can be seen as *low-level feature detectors*. By learning a second layer using as an input the new representation of the data extracted with these low-level feature extractors, it can extract progressively *higher level feature detectors*. This process can continue by adding an extra RBM on the top of the model obtaining newer representations [Hinton et al., 2006]. A representation of this process is depicted in Fig. 2.7.

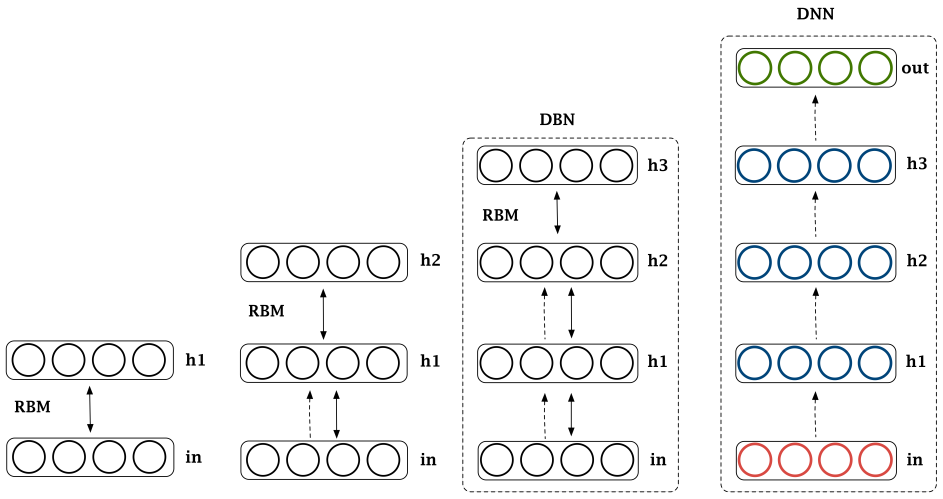


Figure 2.7. Several RBMs are trained independently and stacked to form a DBN. This DBN also serves as a pre-training to a discriminative DNN.

Once the entire DBN is pre-trained layer-wise, it is possible to perform a fine-tuning to adjust all the parameters of the network together according to a criterion. For instance, [Hinton et al., 2006] proposed to use the wake-sleep algorithm ([Hinton et al., 1995]) to fine-tune all the DBN with respect to an unsupervised criterion. Alternatively, it is possible to convert the unsupervised DBN into a discriminative Deep Neural Network (DNN). This process is depicted in Fig. 2.7, and it is performed by adding an extra layer on the top of the DBN. This output layer is composed of units that represent the labels of the discriminative problem, as it has been explained in Section 2.2.1. Note that the weights used to connect this output layer are not pre-trained, but initialized randomly.

[3]Note that [Hinton and Salakhutdinov, 2006] used an unsupervised non-generative model called Auto-Encoder.

Regarding the applications of the DBN model, it has been used as a powerful generative model with many different forms of data in diverse areas like image classification, speech recognition and information retrieval [Dahl et al., 2010; Hinton, 2007; Susskind et al., 2008; Welling et al., 2005]. The recent work present in [Sarikaya et al., 2014] about using DBNs for natural language understanding is also interesting. However, they are mainly used to initialize deep discriminative networks as explained above [Erhan et al., 2010; Hinton et al., 2006; Sermanet et al., 2013]

Even though it is not the topic of this thesis, it is worth to mention the convolutional version of the DBN model (CDBN) [Lee et al., 2009a]. In like manner the standard DBN, the CDBN is built by stacking several convolutional RBMs, and it has been used in image and audio recognition tasks [Huang et al., 2012a; Lee et al., 2009b].

2.3.3 Other unsupervised models

Aside from the models described above, there exist other unsupervised models that have been and are important when talking about DL, but they fall out of the scope of this thesis. For instance, Auto-Encoders [Bengio, 2009] are shallow neural networks that are trained to reconstruct their input using an intermediate representation (code). Auto-Encoders and RBMs are very similar models because they share the same architecture: an input layer that represents the data, a hidden layer that represents the code to be learnt, and the weighted connections between them. However, The Auto-Encoder is trained by minimizing the reconstruction error, so an extra layer is added on the top to represent a reconstruction of the original data. Both sets of weights are *tight*, which means that they are actually the same weights. These models also have the capability of extracting good representations from unlabeled data that work well to initialize deeper networks [Glorot et al., 2010; Rifai et al., 2012].

One of the main problems of the Auto-Encoder model is that it could potentially learn a useless identity transformation when the representation size (the hidden layer) is larger than the input layer (the so-called *over-complete* case). To overcome this potential limitation, there is a simple variant of the model called Denoising Auto-Encoders (DAE) [Vincent et al., 2008] that works well in practice. The basic idea is to feed the encoder/decoder system with a corrupted version of the original input, but to force the system to reconstruct the clean version. This small change allows the DAE to learn useful representations, even in the over-complete case.

Another prominent type of deep unsupervised model is the Deep Boltzmann Machine (DBM) [Hinton and Salakhutdinov, 2012; Salakhutdinov and Hinton, 2009a; Srivastava and Salakhutdinov, 2014]. The DBM is a generative model that contains many layers of hidden variables, and it has no connections between the variables within the same layer. The architecture of the DBM and the DBN models is very similar. The only difference is that the connections between the layers in the DBM are undirected edges, and they are directed in the case of DBNs. This model can also be pre-trained using RBMs [Salakhutdinov and Hinton, 2009a].

Chapter 3

Regularization Methods for RBMs

The first two chapters of the thesis introduced the context of the Deep Learning framework and its most important models and methods. This chapter is focused on one of these models called the Restricted Boltzmann Machine (RBM). More in detail, the chapter describes different regularization techniques for the RBM, and it presents a novel regularization method called Mask Selective Regularization (MSR).

3.1 Introduction

Generally speaking, the regularization concept is based on the idea of an artificial discouragement of complex or extreme explanations of the world even if they fit satisfactorily what has been observed. The motivation behind this topic is that such explanations are unlikely to generalize well to the future, that is, they may happen to explain well a few examples from the past, but this is probably just because of anomalies of these samples.

In the Machine Learning context, the main goal is to learn a model from a set of samples (training data), which is able to generalize well on new and unseen samples (test data). The generalization capability is a key element for the performance of the model, and it is directly related to the complexity of the model itself. On the one hand, if the model is too simple, the solution would be too generic and it will fail to new data because the relations learned between the input variables are too simple. This problem is called *underfitting.* On the other hand, if the model is too complex, it would do much better on the training data; but also it will probably have poor prediction and generalization capabilities, failing indeed on the test data. This usually occurs with complex models with large number of parameters that are able to stick too much to the data. This is the opposite problem and it is called *overfitting.* Obviously, none of these situations is desirable and there are several ways to deal with them. While the underfitting problem is usually quite easy to solve by increasing the complexity of the model, the solution for the overfitting problem is not so trivial. The regularization methods come handy in this case by penalizing non-generic solutions and controlling the model complexity.

As described in Section 2.3.1, the Restricted Boltzmann Machine (RBM) model is an unsupervised generative model that can be represented as a two-layer neural network with the connections between the layers being the parameters of the model. Even though its hidden units can usually represent only binary values, the RBM can learn complex functions thanks to its topology and the large number of these hidden units. This produces a large number of free parameters, which increases the overfitting problem. For this reason, using regularization methods during training often improves the capabilities of this model [Fischer and Igel, 2012; Hinton, 2010].

The rest of the chapter is organized as follows:

- Section 3.2 explains the motivation behind using this idea and the main contributions related.
- Section 3.3 resumes some interesting works concerning the concepts of *sparsity* and *regularization* in the RBM model, specially in computer vision problems.
- Section 3.4 presents and explains in depth a novel regularization scheme called Mask Selective Regularization (MSR).
- Section 3.5 evaluates the method presented and summarizes the results obtained.
- Section 3.6 draws some conclusions about the chapter and with regard to future work.

3.2 Motivation and Contributions

There are several reasons for saying that regularization is an important step during the training process of the RBM. In this process, the weighted connections of the model are adjusted to learn the manifold distribution represented by the input data. In order to avoid the model to stick too much to the data, the regularization technique penalizes these weights to grow too much by adding an extra constrain during training. For this reason, regularization is also known as a *weight-decay* in the RBM's context. As it is stated by Geoffrey E. Hinton in [Hinton, 2010], there are four main reasons for using weigh-decay. The first one has been already explained and attempts to improve the generalization to new data by reducing overfitting. The second one is about the quality of the feature detectors learnt by the RBM. Each set of connections linked to a given hidden unit can be seen as a feature detector, which is activated in presence of interesting parts of the data. In the case of images, these features are usually spatially localized forms around zones called *receptive fields*, forming useful Gabor or Haar-like detectors. However, some hidden units do not learn any useful information. The regularization technique makes the receptive fields of these hidden units smoother and more interpretable by shrinking useless weights. The third reason involves speeding up the training process of the RBM. As it is explained with more details in Section 2.3.1.2, the Contrastive Divergence (CD) algorithm employed to train the RBM model is based on a sampling process that uses a Markov chain. This Markov chain mixes more rapidly with small weights, which means that the method needs less time to be “close” to the target distribution. This is especially important in the CD algorithm because it is based on ignoring the latter steps in the chain [Hinton et al., 2006]. The fourth reason is to *unstick* hidden units that have developed very large weights during training and do not change its state, being always firmly on or off during testing. If a hidden unit is always in the same state regardless of the values of the visible units, this unit is useless because it does not detect any pattern from the data.

All of these reasons enhance the importance of keeping the size of the weights of the RBM under control. The most common way to achieve this is using the so-called L_2 regularization, which imposes the L_2 norm on the weights penalizing large values during training.

A highly related concept to regularization is the concept of *sparsity*. There are two classes of sparsity: sparse activity and sparse connectivity. The former means that only a small fraction of neurons is active at the same time. The later means that each neuron in the hidden layer is connected to only a limited number of neurons in the visible layer. Interestingly, both kind of sparseness have a strong biological inspiration since similar properties have been observed in mammalian brains [Lee et al., 2008]. In this thesis, we have focused on sparse connectivity, although both of them can be easily related. To force sparseness between the layers in the RBM it is necessary that some weights become exactly zero, and this cannot be achieved by using the L_2 regularization previously named. However, by using a different regularization term based on the L_1 norm, it is possible to force real zeros in some connections, that is, to achieve sparse connectivity [Ng, 2004]. The impact of using this kind of regularization is also notable because allows to remove those connections which model weak relations between variables. In a non-sparse RBM, this causes poor performance in presence of noise that directly affects the hidden layer nodes [Tang et al., 2012]. Despite this advantage, the L_1 regularization does not keep the magnitude of the weights under control, and they can grow quite large.

From these premises, we propose a new regularization scheme for RBMs that aims to take into account the advantages of the L_2 and the L_1 regularizations. Therefore, the hypothesis stated suggests that combining both standard regularizations would improve the performance of the RBM model, and most specially, its robustness against noise. The idea is that the two kinds of regularizations should be applied separately using a smart selection method that takes into account the topology of the input space to decide which connections are penalized by each method. This selection is done using a binary mask that changes adaptively according to the state of the network during training. Based on these characteristics, the new regularization proposed is called Mask Selective Regularization (MSR). The new method has been implemented and some experiments on different domains have been evaluated to show its performance and robustness in several classification problems.

3.3 State of the Art

RBMs have been used thoroughly in several tasks with different types of data [Hinton et al., 2006; Hinton and Salakhutdinov, 2009; rahman Mohamed et al., 2009; Salakhutdinov et al., 2007]. The use of different type of visible units is also interesting to better modeling different types of data [Hinton and Salakhutdinov, 2006], as well as the improvement achieved by using rectified linear hidden units to speed up the training process [Nair and Hinton, 2010]. On the other hand, RBMs can also be used as a feature extractor model to feed the new representation of the data into a classifier which is not necessarily a neural network [Cai et al., 2012; Lee et al., 2009c].

Regarding the regularization and sparsity, these two concepts have been analyzed extensively in RBM models. On the one hand, weight-decay method with the L_2 regularization term is commonly added during the training of standard RBMs [Hinton and Salakhutdinov, 2006; Salakhutdinov et al., 2007]. This method has been important not only to avoid overfitting, but also to rise the mixing rate during sampling, as it was demonstrated in [Tieleman, 2008b].

On the other hand, sparsity has been also introduced in several ways. Since the Deep Learning era emerged, some of the works regarding RBM's regularization have been focused on forcing sparse activity. For instance, [Luo et al., 2011] proposed a combined L_1/L_2 regularization upon the activation probabilities of the hidden units to capture the local dependencies among these units. This regularization not only encourages units of many groups to be inactive given observed data, but also makes the hidden units within a group to compete with each other for modeling observed data. However, those groups are fixed by hand and they remain unchanged during training. Another recent article, [Ji et al., 2014], also proposes a sparse RBM model that encourages the hidden units to be sparse through adding a log-sum norm constrain on the totality of the hidden units' activation probabilities. This allows each neuron to learn its optimal sparsity based on the task at a hand, contrary to other methods that constrain the sparsity level of the units to a target value [Lee et al., 2008].

Nevertheless, all of these methods are focused on forcing sparse activity, and the MSR method presented in this chapter is focused on enforcing regularization and sparsity at connections level. Sparse connectivity was introduced many years ago in visual recognition systems that use convolutional networks, where local receptive fields are connected only to a small subset of image pixels [LeCun et al., 1998]. In that case, removing those connections aimed at reducing the number of model parameters and providing a certain amount of translational and scale invariance. However, in the case of the RBMs, the sparse connectivity proposed is not related to those advantages, but rather to improve the performance by making the model more robust agains noise. A similar idea is followed in [Tang and Eliasmith, 2010], where a sparsely connected RBM is trained by assigning a random fixed receptive field to each hidden unit and removing the connections outside this receptive field. Although this method achieves some kind of robustness, the best performance is obtained by applying a probabilistic denoising algorithm that cleans the noisy data a posteriori. The idea of removing connections introduced in this work, inspired partly the creation of our novel MSR method presented in this chapter. However, our assumption is that the removed connections should follow a smarter criterion instead of being chosen randomly and remain unchanged. Also, using square random fields limits the type of data only to images, not being able to apply the method in upper layers of a deeper network.

The idea of removing useless connections has been also touched by [Turcsany and Bargiela, 2014], which share similar ideas with the MSR method. The authors proposed to learn localized features by using biologically inspired Gaussian receptive fields. The removed connections are fixed by a binary mask which is learned dynamically during training. However, the results obtained are only focused on improving the reconstruction error of the RBM.

Another interesting work is presented in [Cho et al., 2012], where a Tikhonov-type regularization for RBMs is studied. The article demonstrates that this regularization can be seen as a combination of the weight decay regularization applied to the connections and the sparsity forced in the hidden activations. However, this combination does not turn out to be significant enough according to the generative and discriminative performance presented in the article.

Aside from the idea of taking advantage of using several types of regularizations, one key concept introduced by the MSR method is to take advantage of the topology of the input space. In this way, [Schulz et al., 2011] proposed an RBM variant that exploits the local nature of the image statistics. The new model employs local connectivity to remove those connections that model long-range dependencies, which are known to be weak in natural images [Huang and Mumford, 1999]. However, it is unclear how to extend this type of sparse connectivity to non-image data where it can not be assumed these local statistics.

Besides these articles, it is also important to cite an alternative regularization and variable selection method called Elastic Net (EN). The EN regularization was originally devoted for linear regression, a lasso convex problem proposed by Zou and Hastie [Zou and Hastie, 2005]. The authors transform the naïve elastic net problem into an equivalent lasso problem on augmented data. Moreover, the naïve elastic net can perform an automatic variable selection in a fashion similar to the lasso. Although it is possible to use the EN approach in RBMs, all the advantages introduced by the authors for the lasso convex problem can not be extended to the RBM due to its non-convexity nature of the optimization problem.

3.4 Mask Selective Regularization for RBMs

This section focuses on the new regularization method proposed for RBMs, called Mask Selective Regularization (MSR). The most important characteristics of this method are explained here, and it is compared to other regularization methods employed in RBMs.

3.4.1 Introduction

Regularization is an important step in any optimization process to prevent overfitting by penalizing complex solutions. In the context of RBMs, the regularization is introduced as a restriction over the parameters during the learning process, as stated before. In such process, the Contrastive Divergence (CD) algorithm (see Section 2.3.1.2) employes a stochastic gradient descent method to minimize a loss function defined by

$$\ell\left(\theta, \mathcal{D}\right) = -\mathcal{L}\left(\theta, \mathcal{D}\right) \ , \tag{3.1}$$

where $\mathcal{L}\left(\theta, \mathcal{D}\right)$ is the log-likelihood of the training set $\mathcal{D}$ (see Eq. 2.11), and θ are the parameters of the model (weights and biases). This loss function can be supplemented by adding an extra term to force some kind of constraint. This term depends on the type of the regularization, as explained down below. For instance, to penalize the magnitude of the weighs it is possible to modify the loss function according to:

$$\ell\left(\theta, \mathcal{D}\right) = -\mathcal{L}\left(\theta, \mathcal{D}\right) + \lambda_2 \|\mathbf{W}\|_2 \ , \tag{3.2}$$

where $\mathbf{W} \in \mathbb{R}^{D \times N}$ is the matrix of the weights that connects the visible and the hidden layers (being D and N the dimensionality (number of units) of these layers, respectively) and λ_2 is a regularization coefficient. According to this expression, the weight-decay method penalizes large weights in $\mathbf{W}$ using the L_2 norm. However, this norm does not force real zeros in weak connections, and this may adversely affect the hidden activation units if there is any kind of perturbation in the visible layer.

A different regularization method is based on using the L_1 norm over the parameters of the model. In the same way, the loss function is modified by adding an extra term:

$$\ell\left(\theta, \mathcal{D}\right) = -\mathcal{L}\left(\theta, \mathcal{D}\right) + \lambda_1 \|\mathbf{W}\|_1 \ , \tag{3.3}$$

where λ_1 is a regularization coefficient. This loss function often causes many of the weights to become exactly zero whilst allowing a few of them to grow quite large. The advantage of the L_1 regularization over the L_2 is that the former performs a real feature selection forcing sparsity in the connections [Ng, 2004]. The feature detectors obtained using the L_1 regularization are strongly localized which eases the interpretation. However, the L_1 norm is not very common in the RBMs' context because perhaps it may remove too many connections and the remaining may contain large weights, which is a known problem for generalization.

One approach in order to combine both regularization terms is the Elastic-Net (EN) [Zou and Hastie, 2005]. This approach applied to the RBM optimization function leads to the expression of the loss function given by

$$\ell\left(\theta, \mathcal{D}\right) = -\mathcal{L}\left(\theta, \mathcal{D}\right) + \lambda_1 \|\mathbf{W}\|_1 + \lambda_2 \|\mathbf{W}\|_2, . \tag{3.4}$$

According to the expression, both regularizations are applied together to all the weights, using the λ_1 and λ_2 parameters to control the contribution of each term.

3.4.2 A Loss function combining L_2 and L_1

Each of the L_1 and the L_2 regularizations has its own advantages and disadvantages, so we propose to define a new regularization scheme that could combine smartly the advantages of each norm into a single framework. The main idea is to adaptively split the set of weights into two disjoints sets, so that the L_1 regularization is applied on one set and the L_2 on the other one. A new loss function is defined, and it is given by

$$\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D}) + \lambda_1 \left\| \mathbf{W} \circ \hat{\mathbf{R}} \right\|_1 + \lambda_2 \left\| \mathbf{W} \circ \mathbf{R} \right\|_2 , \tag{3.5}$$

where $\mathbf{R}$ is a $D \times N$ binary mask and $\hat{\mathbf{R}}$ is its bit-wise complementary mask. Since $\mathbf{W}$ and $\mathbf{R}$ are multiplied element-wise, the ones in $\mathbf{R}$ represent the elements in $\mathbf{W}$ where the L_2 regularization is applied. Similarly, the ones in $\hat{\mathbf{R}}$ correspond to the elements in $\mathbf{W}$ where the L_1 regularization is used. This loss function enforces many elements of $\mathbf{W}$ to be exactly zero (L_1 norm) and, at the same time, avoids the remaining ones to grow too large (L_2 norm). In order for this to be meaningful, the matrix $\mathbf{R}$ must be responsible for taking into account the topology of the input space to decide smartly which connections are penalized by each method. This type of regularization is called Mask Selective Regularization (MSR).

It is important to note that, although both the EN regularization and the MSR employ the L_1 and the L_2 regularizations, there is a significant difference between them. In MSR either the L_1 or the L_2 is applied to each parameter, whereas in EN both regularizations are applied together to all the parameters. Typically, EN is unfruitful to combine L_1 and L_2 in cases where λ_1 and λ_2 have different magnitude orders. For instance, a large λ_1 dominates the minimization of the loss function towards a sparse solution, while a large λ_2 focuses towards a shrinkage solution. However, this undesirable effect is avoided with the selective regularization MSR because λ_1 and λ_2 values are applied separately over different weights.

3.4.3 Mutual information and Correlation coefficient

Before going into detail about the MSR method, it is necessary to define a measure capable of taking into account the relations between the input variables to build the $\mathbf{R}$ matrix. This measure should be able to detect the strong and the weak dependencies from the input topology.

Assuming that the input representation space is formed by random variables (for instance, pixels in the case of images), the Mutual Information (MI) of these variables is a measure of the their mutual dependence. Formally, the MI of two discrete random variables X and Y can be defined as

$$I(X, Y) = \sum_{y \epsilon Y} \sum_{x \epsilon X} p(x, y) \log \left(\frac{p(x, y)}{p(x)\, p(y)} \right) , \tag{3.6}$$

where $p(x, y)$ is the joint probability distribution of X and Y, and $p(x)$ and $p(y)$ are the marginal probability distribution functions of X and Y respectively. The MI measure can also be interpreted as the predictability of the second variable by knowing the first one, or the stored information in one variable about the other variable.

However, the input units of the RBM model represent probabilities before the sampling process. It is non-trivial to estimate the MI for continuous data because the sums in Eq. 3.6 become a double integral. For this reason it is common to approximate the MI measure by the Pearson's correlation coefficient, which is used to measure the linear dependence between random variables [Li, 1990; Song et al., 2012]. Formally, the correlation coefficient between two random variables X and Y is defined as

$$corr\,(X,Y) = \frac{cov\,(X,Y)}{\sigma_X \sigma_Y} = \frac{E\left[(X-\mu_X)(Y-\mu_Y)\right]}{\sigma_X \sigma_Y} \ , \tag{3.7}$$

where μ_X and μ_Y are the expected values of the random variables, and σ_X and σ_Y are their standard deviations, respectively.

Although MI is a more general measure to evaluate the relation between variables, in this thesis we use the Pearson's correlation coefficient as a surrogate measure of the MI. The correlation coefficient has the advantage of being straightforward to compute for continuos data, unlike the MI. Also, it is assumed that capturing the linear relation between the input variables is enough to mark out and define the connections where each regularization should be applied. The next section explains how the binary regularization mask $\mathbf{R}$ is built by computing these correlations.

3.4.4 The binary regularization mask

The first step to be solved by the MSR approach is how to build the binary matrix $\mathbf{R}$. This matrix should fulfill two characteristics. On the one hand, $\mathbf{R}$ should take into account the topology of the input data: the weak dependencies between the visible units should be regularized using the L_1 norm, and stronger dependencies should be regularized using the L_2. On the other hand, it also might be appropriate that this mask could change adaptively to accommodate the current state of the weights during the training process. For now, just assume that $\mathbf{R}$ is built dynamically as a part of the training process by selecting many times a subset of binary vectors from a precomputed set $\mathcal{M}$.

Let us focus on the $\mathcal{M}$ set. This set is formed by D different binary vectors (masks) $\mathbf{m_i} \in R^D$, where each $\mathbf{m_i}$ mask is related to a different visible unit. Therefore, there are as many unique binary masks as number of visible units. The goal of these masks is to sweep the most influence areas of each dimension in the training set. The areas defined by the masks are calculated using the Pearson's correlation coefficient between all the input dimensions across the entire training set. Therefore, each binary element m_{ij} of the vector $\mathbf{m_i}$ would be 1 if the correlation between the visible units i and j is high, and 0 otherwise. More formally, to build the binary vector related to the unit i, i.e. $\mathbf{m_i}$, its c components with the highest correlation values are set to 1, and the remaining ones are set to 0. In order to decide the c parameter it is proposed to keep the correlation ratio above a certain threshold α, which can range from 0 to 1. Mathematically, this criterion fulfills the equation

$$\frac{\sum\limits_{j \in \mathcal{S}} |corr\,(i,j)\,|}{\sum\limits_{j=1}^{D} |corr\,(i,j)\,|} > \alpha \ , \tag{3.8}$$

where $\mathcal{S}$ is the set of indexes of the highest c correlation values, and α is a tuneable parameter of the model.

Fig. 3.1 displays the main steps to precompute the $\mathcal{M}$ set according to what has been stated above. This figure represents the process followed for the MNIST handwritten digits dataset[1], which is specially suitable to visually understand the process. According to the figure bellow, the first step is to compute the correlation coefficient using Eq. 3.7 between all possible combinations of pairs of pixels from the input space across the entire training set. All these values can be represented as a grayscale symmetric matrix where white color denotes high correlation and black color denotes low correlation. Note that this representation is given by the absolute value of the correlation coefficient. From this matrix, each binary mask $\mathbf{m_i}$ of the $\mathcal{M}$ set is represented as a column vector, where the indexes that satisfy Eq. 3.8 are set to 1. In other words, each of these vectors activates their components if the current input unit is *correlated* with respect to the others. Note that the $\mathcal{M}$ set is represented as a binary matrix in the figure by appending all the vectors.

The binary masks precomputed following the process explained depend greatly on the α parameter. In order to visually assess the effect of this parameter, Fig. 3.2 represents the resulting binary mask related to a given visible unit varying the α parameter. These images are obtained by rearranging the elements of the binary vectors to form a square matrix. Note that all the masks are associated to the same visible unit, which is spatially localized by a red dot. In the case of images, small values of α generates masks that are localized around the visible unit. Larger values of α capture more long-term dependencies and can relate areas that are not spatially connected.

Once how the α parameter affects the size of the binary mask is clear, it is also interesting to visually assess the regularization masks related to different input units fixing α. Fig. 3.3 shows a few examples of these regularization masks obtained from the MNIST dataset using $\alpha = 0.7$. Each mask intends to capture the strongest (in white color) and weakest (in black color) dependencies for each visible unit. In the specific case of the MNIST database, it would be 784 precomputed masks similar to those represented in the figure because there are 28×28 visible units, which is the size of the images.

At this point, it is important to recall that these masks are created without assuming any knowledge about the geometry of the data, unlike other methods that force sparse connections by hand assuming the image layout of the samples [Müller et al., 2010; Tang and Eliasmith, 2010]. For this reason, this method extends its use to non-image data as well without any modification. Actually, this hypothesis has been demonstrated in some experiments performed with text data (see Section 3.5.4). In this regard, these binary masks can be obtained not only in the first RBM of a deep network, but also in upper layers where it cannot be assumed that the data follows any particular layout.

After explaining how to precompute the binary vectors for the $\mathcal{M}$ set, the next step explains how the regularization mask R, which is applied to the weights, is built dynamically by means of selecting a subset of these vectors. Next section explains this process.

[1]See Appendix A.1

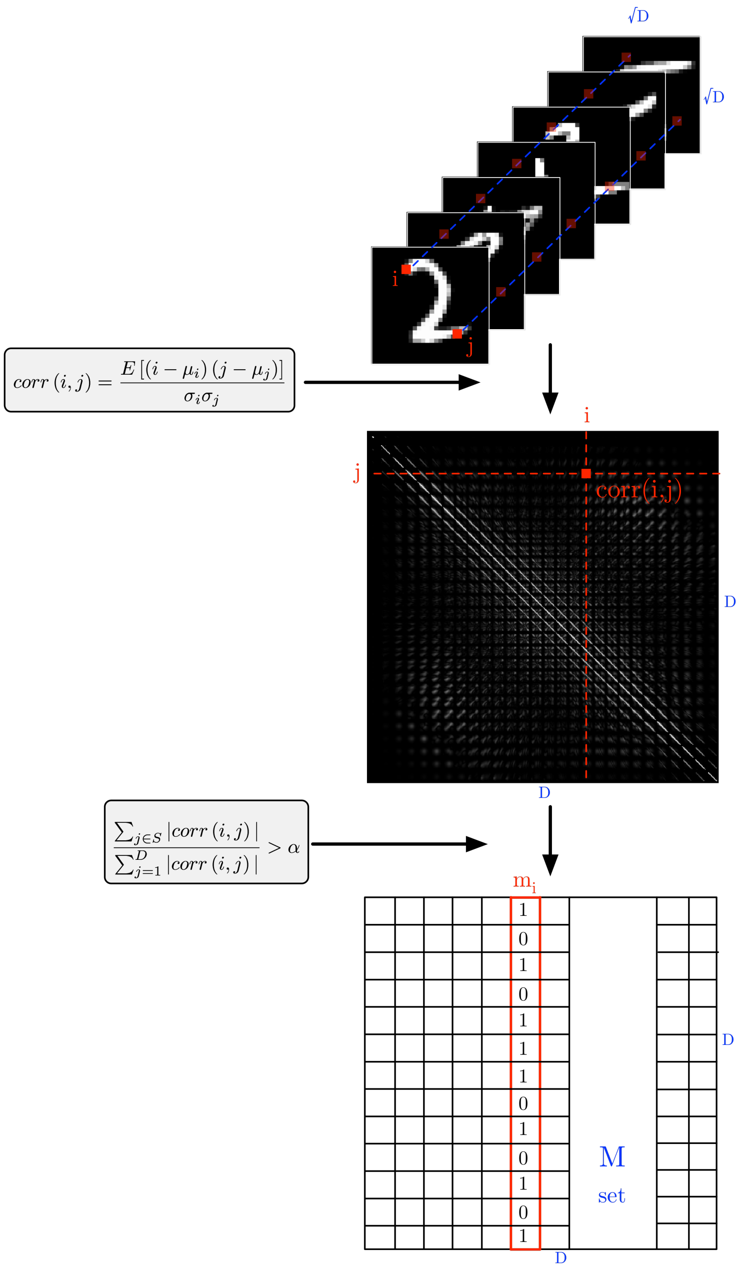

√D
√D
i
j
$corr\left(i,j\right)=\frac{E\left[\left(i-\mu_i\right)\left(j-\mu_j\right)\right]}{\sigma_i\sigma_j}$
i
j
corr(i,j)
D
D
$\frac{\sum_{j\in S}\left|corr\left(i,j\right)\right|}{\sum_{j=1}^{D}\left|corr\left(i,j\right)\right|}>\alpha$
m_i
1 0 1 0 1 1 1 0 1 0 1 0 1
M
set
D
D

Figure 3.1. Graphical representation of the process to precompute the binary masks of the $\mathcal{M}$ set for the MNIST dataset.

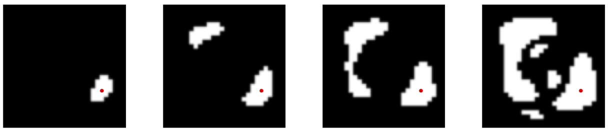


Figure 3.2. Binary masks for $\alpha \in \{0.2, 0.4, 0.6, 0.8\}$ for the MNIST dataset. The red dots indicate the visible units related to each binary mask.

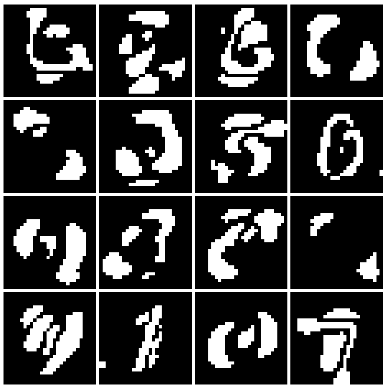


Figure 3.3. Some examples of binary regularization masks for the MNIST dataset.

3.4.5 Topology selection and convergence

Let us remember that the matrix $\mathbf{R}$ is used to selectively apply a different regularization term over the weight matrix $\mathbf{W}$, and it is built by means of selecting a subset of masks from $\mathcal{M}$. Let $\mathbf{w}_j$ and $\mathbf{r}_j$ be the j^{th} column of the matrix $\mathbf{W}$ and $\mathbf{R}$ respectively. Each element of $\mathbf{w}_j$ can be regularized using either the L_2 or the L_1 norm according to the binary values of $\mathbf{r}_j$. Each $\mathbf{r}_j$ is selected among the precomputed masks $\mathcal{M}$ using an energy criterion given by:

$$\mathbf{r}_j = \max_{\mathbf{m}_k \in \mathcal{M}} E_{in}^k - E_{out}^k \tag{3.9}$$

$$E_{in}^k = \frac{\mathbf{m}_k^\intercal \mathbf{w}_j^2}{||\mathbf{m}_k||_1} \tag{3.10}$$

$$E_{out}^k = \frac{\hat{\mathbf{m}}_k^\intercal \mathbf{w}_j^2}{||\hat{\mathbf{m}}_k||_1} \tag{3.11}$$

where $\mathbf{m}_k$ is a binary mask from the precomputed $\mathcal{M}$ set, and $\hat{\mathbf{m}}_k$ is its bit-wise complementary mask. The E_{in}^k value is the *positive energy* of the mask $\mathbf{m}_k$ with the feature $\mathbf{w}_j$ averaged by the mask's area. The intuitive idea behind this equation is that, given a feature vector $\mathbf{w}_j$, E_{in}^k will be high if the large weights of $\mathbf{w}_j$ mostly overlap with the mask $\mathbf{m}_k$. Similarly, E_{out}^k value is the *negative energy.* This term penalizes the case where large values of $\mathbf{w}_j$ are outside the mask $\mathbf{m}_k$. Therefore, according to this criterion, $\mathbf{r}_j$ is selected as the binary mask $\mathbf{m}_k \in \mathcal{M}$ that best "matches" with $\mathbf{w}_j$.

One thing to keep in mind is the fact that the selection process is done independently for each feature detector $\mathbf{w}_j$, so the same binary mask from $\mathcal{M}$ can be selected many times. In other words, there could be repeated masks in $\mathbf{R}$ because the selection process is not exclusive. Obviously, this fact will happen when $n > d$. However, this effect may be desirable because some local areas would require several hidden units with the same mask in order to explain all the variability that appears in that portion of the representation space.

For a better understanding of how this energy criterion works, Fig. 3.4 displays some of the features learned at the end of the training process for the MNIST dataset, and their binary masks associated overlaid in red color. Note that each $\mathbf{w}_j$ vector links all the visible units to the j^{th} hidden unit, so it is possible to display each feature detector as an image by resampling the vector to a square matrix. Note also that the black color in the image denotes positive values, the white color denotes negative values, and the gray color denotes values near zero. According to the figure, the goal

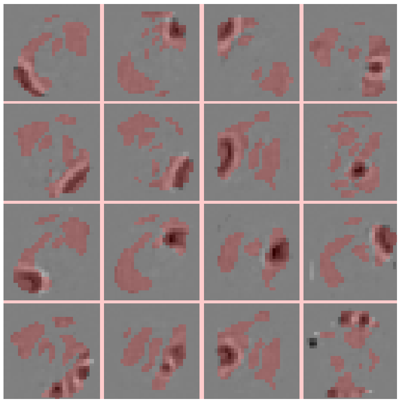


Figure 3.4. Learned features for the MNIST dataset along with their corresponding binary masks overlaid in red color.

of this criterion is clear: to spatially localize the features learned inside the binary masks, so that the L_2 norm is applied to the feature itself and the L_1 norm is applied outside its influence area to force real zeros.

Regarding the training process of the RBM (see Section 2.3.1.2) using MSR, the regularization mask $\mathcal{R}$ is changed according to the state of the network during the iterative process. Therefore, the selection criterion given by Eq. 3.9 is applied several times and the mask selected by each unit is probably going to change during the training Actually, the binding between the regularization masks and the weights is quite changing at the beginning of the process. At early epochs, since the values of the weights are initialized randomly, the feature detectors are not well defined yet and the binary masks that best fit the weights vary significantly. As the process evolves, the correspondence freezes and each feature detector gets associated with a fixed regularization mask. To ensure the convergence of the method, a probability assigned to the capacity of changing the current mask at each step is defined. This probability decreases linearly with the training time, so the chance for changing the mask in very initial iterations is high, but tends to zero for the last iterations. Therefore, if the energy criterion selects a different binary mask, this change will be made depending on a probability given by the ratio between the elapsed steps at that moment an the total training steps. Otherwise, the mask remains unchanged. This decision is made by sampling from a Bernoulli distribution with the probability previously defined.

3.4.6 The MSR Algorithm

The steps of the MSR method described in the previous section can be integrated into the CD algorithm to train the RBM model. The loss function to be minimized is defined in Eq. 3.5. The derivative of that loss function with respect to the parameters of the model yields a very simple learning rule given by

$$\Delta w_{ij} = \epsilon\big(\langle v_i h_j\rangle_{data} - \langle v_i h_j\rangle_{model} - \lambda_1 sgn\left(w_{ij}\cdot\hat{r}_{ij}\right) - 2\lambda_2\left(w_{ij}\cdot r_{ij}\right)\big) \ , \tag{3.12}$$

where the last two terms are the derivatives of the L_1 and the L_2 regularization terms respectively, and $sgn(x)$ is the *signum* function of x. Note that the update rule for the bias remains unchanged because the regularization does not apply to this parameter. This expression is usually evaluated every time after estimating the gradient on a set of samples called *batch*, which is a common practice in gradient-based methods to get a smoother convergence. However, the update of the **R** matrix in the MSR method is performed only once at every *epoch*, which comprises an entire pass over the training set. Therefore, there are several weighs' updates between each MSR iteration. This is important because it allows the weights to settle to stable configurations between each epoch and reduces the computational cost.

The most important steps of the MSR method within the CD algorithm are summarized in Fig. 3.5. This process starts by precomputing the $\mathcal{M}$ set, obtaining several binary masks that sweep the most influence areas from the training set. After that, the binary matrix **R** is built dynamically in each epoch by selecting the masks from $\mathcal{M}$ that best matches the current weights' states according to the energy criterion defined in Eq. 3.9. The figure depicts how the values of w_j gradually evolve towards a useful feature detector. In the same way, r_j also changes by adapting itself to the state of the weights. At the end of the training process, the RBM will learn useful feature detectors regularized with the L_2 norm on its influence area and with the L_1 norm applied outside.

To sum up the characteristics of the MSR algorithm, it is important to underline two key points of this method. First, this algorithm does not make any assumption a priori about which regularization masks should be used. This selection is guided only in accordance with the topology of the data and the current values of the parameters of the model. In fact, we have observed that there exist masks in $\mathcal{M}$ that are never used while others are reused several times in different feature detectors. Another interesting point is that MSR serves as a guide to indicate where coefficients should be enforced

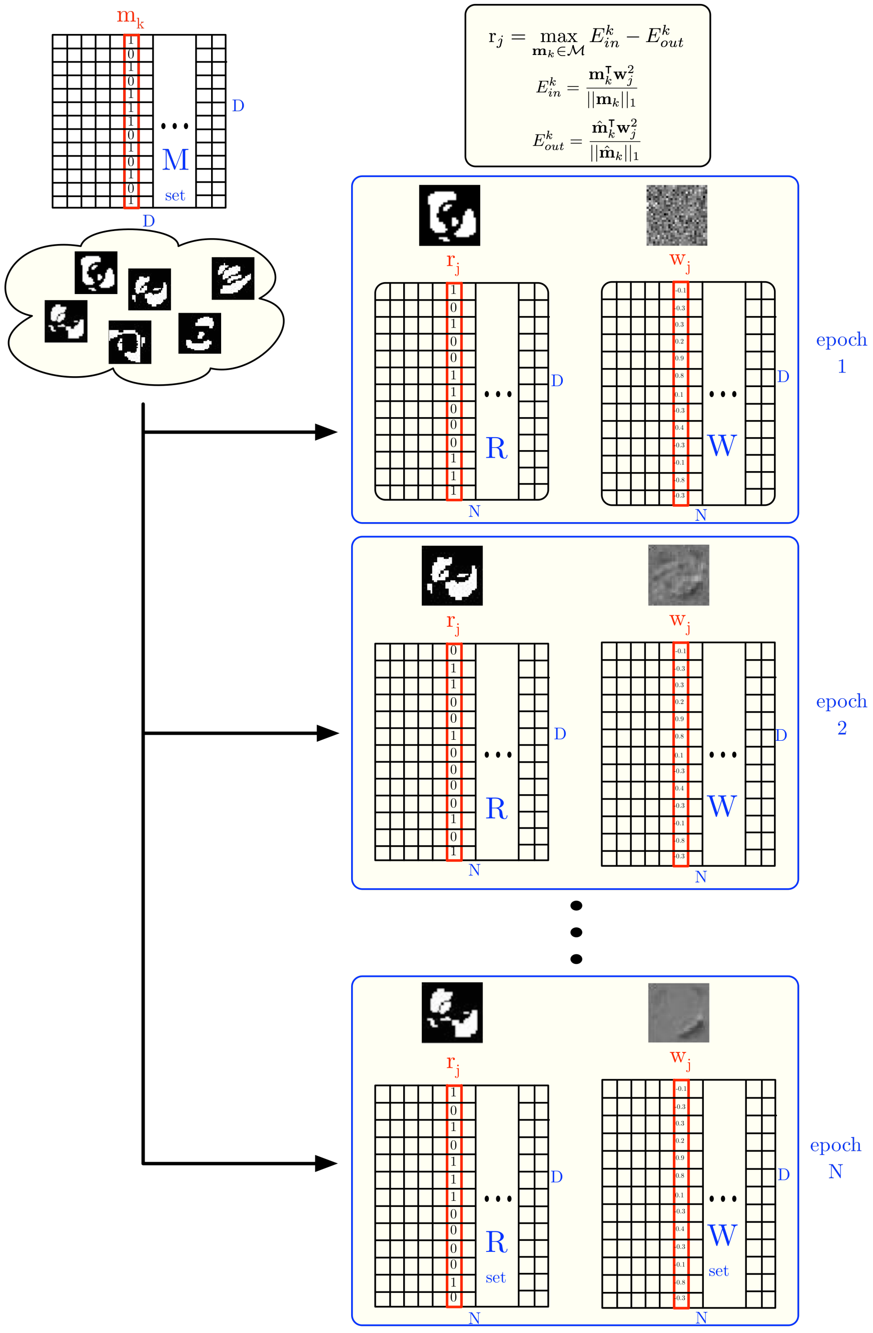

m_k
M
set
D
D
$\mathrm{r}_j = \max_{\mathbf{m}_k \in \mathcal{M}} E^k_{in} - E^k_{out}$
$E^k_{in} = \frac{\mathbf{m}_k^\intercal \mathbf{w}_j^2}{||\mathbf{m}_k||_1}$
$E^k_{out} = \frac{\hat{\mathbf{m}}_k^\intercal \mathbf{w}_j^2}{||\hat{\mathbf{m}}_k||_1}$
r_j
w_j
R
W
N
D
epoch 1
epoch 2
epoch N
set

Figure 3.5. Graphical representation of the MSR method within the CD algorithm.

to be zero. However, this guide is flexible because there may exist situations where higher order relationships between visible units are not captured by the binary masks. In this case, the feature related to those visible units may have non-zero values in the L_1 regularized weights, i.e. outside the mask. The algorithm allows these cases. This fact is illustrated in the bottom-right feature shown in Fig. 3.4, where large weights represented by a black dot lay outside the mask. Obviously, this phenomenon depends on the intensity of each regularization term in the minimization process.

3.5 Experiments

3.5.1 General protocol

This section describes the general protocol for the evaluation carried out regarding the different regularization schemes described before and comparing the results with the MSR model presented. We propose an evaluation over different datasets with image and non-image data. First, two popular handwritten digits datasets are considered: the MNIST and the US Postal Service (USPS). Second, in order to show the capabilities of the MSR to model the topology of the input space from non-image data, we carried out also some experiments using the 20-Newsgroups text classification dataset. Finally, we have used the CIFAR-10 to evaluate our algorithm in a more complex scenario that contains natural color images. These datasets and their evaluation method are described in Appendix A.

All of these experiments have a common protocol described below. First of all, an RBM model is trained including one of the regularizations proposed using the CD_1 algorithm. In some experiments, the model employed becomes deeper by stacking several RBMs on the top, obtaining a DBN model as explained in Section 2.3.2. The training of each RBM is done independently in a complete unsupervised way using the entire training set of the database. After that, the RBM/DBN network is converted into a discriminative model by adding an extra layer that represents the labels of the samples. The entire network is fine-tuned using the Backpropagation algorithm minimizing the cross entropy error on the output layer. The fine-tuning process stops when the average cross-entropy error on the training data fall bellow a pre-specified threshold. To fix this threshold value, we fine-tune the network using only a subset of the training samples and using the remaining ones as a validation set (around the 20% of the training set). The cross-entropy threshold value is fixed with the fewest classification error on the validation set.

The goal of the protocol described is to evaluate how the regularization method applied to the RBMs during the pre-training affects the discriminative performance of the resulting network after the fine-tuning. In other words, we use the RBM model as a feature extractor and we want to evaluate the quality of these features as initialization of a discriminative networks. To this end, five regularization schemes have been compared:

- **No REG**: this model corresponds to an RBM trained without any regularization penalty.

- **L_2**: this model corresponds to an RBM trained with just the L_2 regularization.
- **L_1**: this model corresponds to an RBM trained with just the L_1 regularization.
- **EN**: this model corresponds to an RBM trained with the Elastic Net configuration, that is, using the L_2 and the L_1 together over all the weights.
- **MSR**: this model corresponds to an RBM trained with the new MSR method proposed.

In all cases, a model selection is needed in order to obtain the best parameter configuration in each case. For the L_1 and the L_2 methods, this model selection performs a search over λ_1 and λ_2 among the values 0.0001, 0.001 and 0.01, fixing each value according to the best discriminative performance on the validation set in each case. For the EN method the grid search is performed jointly over λ_1 and λ_2. Finally, for the MSR method the grid search includes also the α parameter in the range $0.3 - 0.8$ with a step of 0.1. It is important to make it clear that the regularization process is only applied during the pre-training step. Once the models have been pre-trained, the discriminative fine-tuning procedure is the same in all cases and does not force any restriction. Also, in the case of the DBN model it is important to highlight that the regularization is applied in all the stacked RBMs, including the MSR approach which is able to work in these upper layers. Actually, some extra experiments on the MNIST dataset have been performed to show the advantage of using MSR not only in the first layer, but also in upper layers of the DBN model.

Noisy data

In order to assess the robustness of the MSR approach, some extra experiments over the handwritten digit databases have been performed to evaluate the performance not only on the clean test data, but also on a noisy version. To this end, and inspired by [Tang and Eliasmith, 2010], the original test partition has been corrupted with three kinds of noise to reflect some possible sources of error. The first source of error is a random noise where 10% of the pixels are randomly activated. The second one introduces a border of two pixels wide to the images. Finally, the third one simulates a block occlusion by adding a square to the images in a random location. The area of that square is set to be one sixteenth of the total area of the image. We can see an example of each kind of noise in Fig. 3.6. It is important to mention that, in these experiments, the models are trained and validated in absence of noise in all cases. Similar experiments over a noisy test set are performed on the 20-Newsgroups text classification dataset where the noise is obtained by modifying the word counts.

3.5.2 Experiments with MNIST

The MNIST is a database[2] of handwritten digits with 10 different classes. The pixel values are normalized to the range $[0, 1]$ and these values are considered probabilities for the binary input units. In these experiments we have employed the same $784 - 500 - 500 - 2000 - 10$ deep network used by Hinton [Hinton and Salakhutdinov,

[2] For further information about MNIST, see Appendix A.1

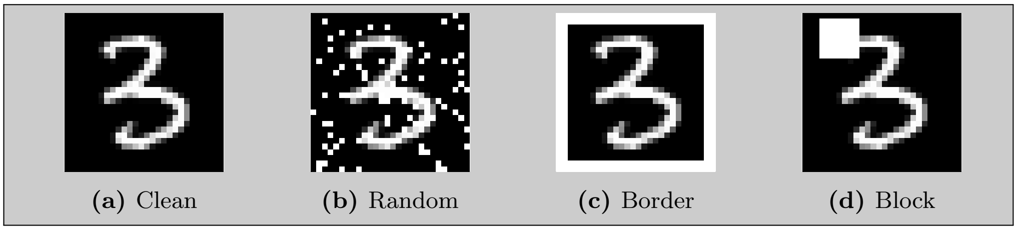


Figure 3.6. Clean and noise images

2006], which achieves 1.14% error on the test set. This notation means that the deep unsupervised network used as a pre-training is formed by three stacked RBMs with 500, 500 and 2000 hidden units respectively. The input layer has 784 visible units, each one representing one dimension of the input data. To build a discriminative network, 10 softmax units are added at the end of the network, each one representing one possible label. During the pre-training each RBM has been trained for 50 epochs. Weights were initialized with small random values sampled from a normal distribution with zero mean and standard deviation of 0.1. The learning rate value was set to 0.1 for both weights and biases. The regularization terms λ_1 and λ_2 were fixed by model selection to 0.0001 in all cases, whereas α is fixed to 0.7 in the case of the MSR.

Quantitative results

To quantitatively evaluate the methods proposed, some results are given in Table 3.1. Note that in this table we have included not only the results about performing the fine-tuning with all the training samples, but also with a small fraction of them. This evaluation aims to simulate the situation where we have tons of unlabeled data but scarce labeled examples. Actually, this is a common situation in most real problems. If we make the assumption that the unlabeled data can be labeled with the same labels as the classification task, we can address the problem within a semi-supervised framework where all the training data is used for the unsupervised pre-training and a small labelled fraction is employed for the fine-tuning. According to the results, we observe that our model MSR achieves the lowest error rate in almost all cases with noisy data. With this kind of corrupted data, it is clear the need of using some method that involves the L_1 penalty to force real zeros in the connections. In contrast, the RBMs trained without regularization or with the L_2 are severely affected by the noise in all its forms. On the other hand, these methods together with the MSR approach perform reasonably well on clean data. In fact, MSR obtains a 1.05% error rate in the clean test set, which is comparable to other results published for the permutation-invariant version of the MNIST task using similar networks. [Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2009a; Tang and Eliasmith, 2010].

We have also conducted an extra experiment on this dataset to show the advantage of using MSR in the upper layers of the deep network. These results are summarized in Table 3.2, where the classification error rate is shown for the same network of the previous experiment depending whether or not the MSR algorithm has been applied to each RBM. Note that the first and the last rows of this table correspond to the results

Table 3.1. Error rate (%) on the MNIST test set for both clean and noisy images using different regularization schemes and varying the number of labeled samples used in the fine-tuning.

Num. samples	**Model**	**Error rate (%)**			
		Clean	**Random**	**Border**	**Block**
100	No reg	**16.30**	39.85	86.72	24.56
	L_2	16.81	20.72	76.12	24.64
	L_1	18.58	20.95	21.46	24.49
	EN	17.63	19.38	21.76	23.83
	MSR	16.37	**18.30**	**19.80**	**22.64**
500	No reg	**6.52**	34.16	84.24	18.13
	L_2	6.61	11.90	70.12	17.74
	L_1	7.59	11.54	9.50	16.15
	EN	7.83	11.16	10.10	16.00
	MSR	6.59	**9.74**	**8.66**	**15.20**
1000	No reg	**4.91**	34.52	83.74	17.24
	L_2	5.03	10.77	71.68	16.89
	L_1	5.62	10.34	7.83	13.82
	EN	5.75	10.38	8.60	14.81
	MSR	4.99	**8.77**	**7.60**	**13.72**
All	No reg	1.11	1.39	87.61	16.46
	L_2	1.07	1.20	79.86	16.24
	L_1	1.11	1.21	**2.82**	12.83
	EN	1.16	1.24	3.37	13.17
	MSR	**1.05**	**1.13**	2.89	**12.44**

already presented in Table 3.1. According to these results the best performance is

Table 3.2. Effect of applying MSR to different layers of the network. Error rate (%) on the MNIST test set for both clean and noisy images.

Layers			**Error rate (%)**			
1st	**2nd**	**3rd**	**Clean**	**Random**	**Border**	**Block**
No Reg	No Reg	No Reg	1.11	1.39	87.61	16.46
MSR	No Reg	No Reg	1.14	1.25	4.37	13.08
MSR	MSR	No Reg	1.09	1.19	5.34	12.73
MSR	MSR	MSR	**1.05**	**1.13**	**2.89**	**12.44**

achieved when MSR is used in all layers. Also it can be seen a gradual improvement achieved when more layers are regularized with the MSR method. Furthermore, these results demonstrate that MSR also works when the topology of the data is unknown as in the case of the upper layers of the network.

Qualitative results

Besides the numeric results presented that show the capabilities of the MSR approach, it is also interesting to report other qualitative measures to check the effect of using MSR instead of other regularizations. On the one hand, Fig. 3.7 shows the histogram of the weighs in the first RBM using the regularization schemes presented. The histogram without any regularization is also included in the figure for completeness. It can be seen that MSR inherits properties from both L_2 and L_1 regularizations: it forces many coefficients to be exactly zero like in L_1 and prevents the coefficients to grow too large like in L_2. The histogram of the EN regularization is also interesting since it is quite similar to that of the MSR. Obviously EN uses both regularizations, but it is clear according to the numeric results previously reported that applying both to the same weight is more difficult to take advantage of their potential.

On the other hand, another qualitative visual assessment can be obtained comparing the features learned by the first RBM model. Fig. 3.8 compares some features learned on the MNIST dataset without any regularization and with the MSR algorithm. According to the images it is clear that MSR obtains cleaner features due to the effect of the L_1 which forces zeros outside the influence area of each feature. We have not include the other methods because the differences are not so significant.

3.5.3 Experiments with USPS

The USPS[3] dataset is composed of images of handwritten digits with 10 classes, similar to the MNIST. The pixel values are normalized to the range $[0, 1]$ and these values are considered probabilities for the binary input units. For these experiments we have used a $784-300-300-1200-10$ deep network, which is the same architecture as in the MNIST but using only three-fifths as many units in each hidden layer, as suggested by [Hinton et al., 2006]. The training and testing procedure and the value of the other parameters are the same as in the MNIST, except that the model selection method fixed the parameters $\lambda_1 = 0.001$ and $\alpha = 0.6$ for the MSR method. The results are displayed in Table 3.3. According to these results, the MSR outperforms

Table 3.3. Error rate (%) on the USPS test set for both clean and noisy images using different regularization schemes.

Model	**Error rate (%)**			
	Clean	**Random**	**Border**	**Block**
No reg	5.17	33.45	71.08	28.36
L_2	5.10	28.26	62.78	28.55
L_1	5.24	27.18	62.63	26.75
EN	5.75	**20.82**	59.00	24.55
MSR	**5.03**	23.16	**55.64**	**24.25**

the other regularizations on the clean version of the test data, obtaining a 5.03% error rate. This result is similar to others obtained also with neural networks [LeCun et al.,

[3]For further information about USPS, see Appendix A.2

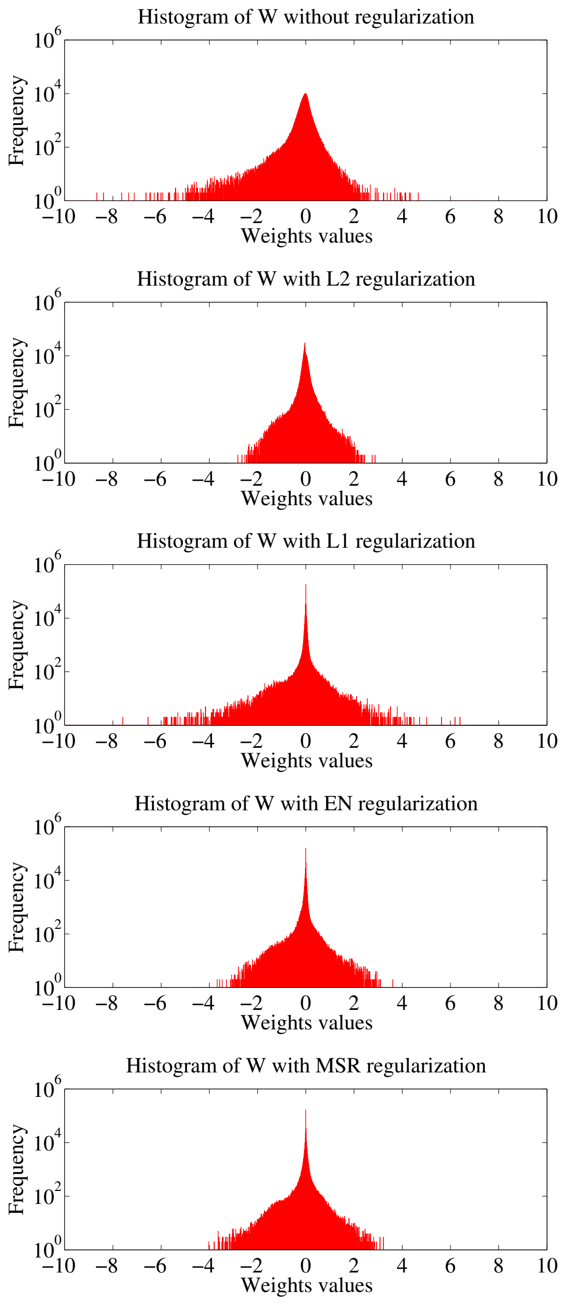
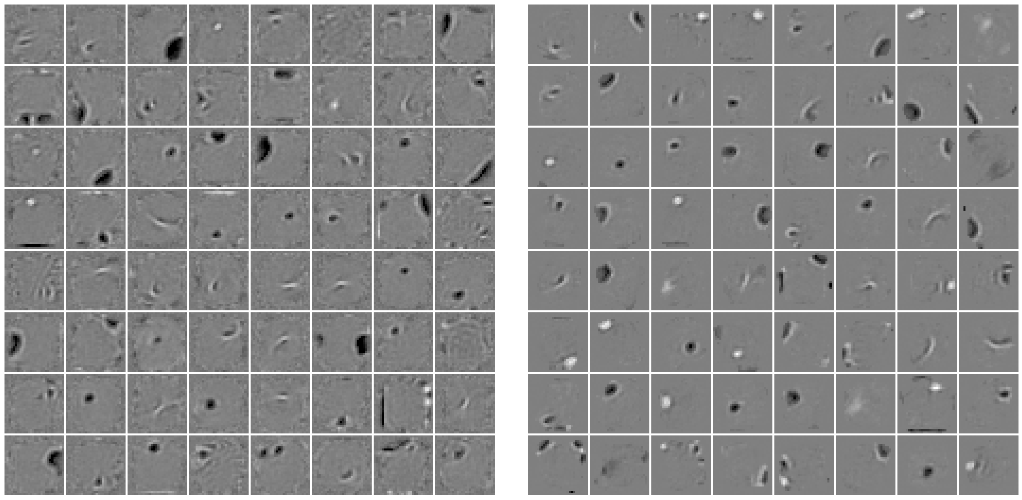


Figure 3.7. Histogram of weights for different regularization schemes in the first RBM for the MNIST dataset.


a) Some features learned by the RBM without any regularization

b) Some features learned by the RBM with the MSR regularization.

Figure 3.8. Comparison of a random selection of features learned by the RBM in the first layer.

1990], but a bit higher than the state-of-the-art in the same database [Jiang et al., 2014b; Yang et al., 2011]. On the other hand, there is also a clear advantage of the MSR method when dealing with noise in most cases. It should be mentioned that, since USPS images are smaller than the MNIST images and their digits are scaled to fit the available area, the Border and Block noises affect the error rate more severely in the USPS case compared to the MNIST dataset.

3.5.4 Experiments with 20-Newsgroups

The 20-Newsgroups corpus[4] is a text dataset with 20 different classes. The word count values of the data are scaled to the range $[0, 1]$ to simulate probabilities. As it was done in the experiments with the handwritten digits, we would like also to evaluate the regularization methods with noisy data. To simulate the noise we have added randomly word counts to each data vector, varying the percentage of the total words corrupted between 10% and 30%. The network employed is formed by just one RBM with 1000 hidden units, so, following the nomenclature, the architecture of the discriminative network is $5000 - 1000 - 20$. Both λ_1 and λ_2 were fixed to 0.0001 in all cases, and α was set to 0.6 in the MSR case. The training and testing procedure and the value of the other parameters are the same as in the handwritten digits experiments. Table 3.4 shows the results. According to the results the RBM trained without any regularization and the MSR model perform best on the clean data. The 22.13% error rate obtained with MSR is comparable, even a bit lower, to

[4]For further information about 20-Newsgroup, see Appendix A.3

Table 3.4. Error rate (%) on the 20-Newsgroups test set for both clean and noisy data.

Model	**Error rate (%)**			
	Clean	**10%**	**20%**	**30%**
No reg	**22.11**	24.14	25.56	28.09
L_2	22.23	23.85	25.17	**27.45**
L_1	22.35	24.32	26.38	28.94
EN	22.73	24.38	26.57	28.74
MSR	22.13	**23.73**	**25.14**	27.48

other results obtained using similar models [Larochelle and Bengio, 2008]. In presence of noise, MSR also obtains the best results in almost all cases, despite the differences are not as significant as in the other tasks. Probably this is due to the very nature of the data representation, which is already very sparse and it is hard to tell the effect of the L_1 regularization. An additional research about studying more suitable representations for this kind of data would be interesting. It also should be evaluated the use of other type of visible units which better deals with the sparseness of the data. Besides this, it is worth to mention that we have found the inability of the EN to deal with values of λ_1 and λ_2 of different magnitude orders, which worsens the results extremely in this task. This effect does not happen using MSR, which gives reasonable results even in this extreme case.

3.5.5 Experiments with CIFAR-10

The CIFAR-10 dataset[5] represents 10 classes of natural scene images. The RBM employed for the unsupervised pre-training has one hidden layer with 4000 units. The pixel intensities of the natural images of this dataset are real-valued data, so the standard binary visible units of the RBM are not the best way to model them. To deal with this kind of data we replace the visible binary units with linear units with Gaussian noise, as described in Section 2.3.1.3. Each component of the data is normalized to have zero mean and unit variance. We train the GRBM model in the same way using the CD_1 algorithm. Weights were initialized with small random values sampled from a normal distribution with zero mean and standard deviation of 0.05. The learning rate value was set to 0.001 for both weights and biases. The regularization terms λ_1 and λ_2 were set to 0.001 for all the models, whereas α is fixed to 0.3 for the MSR model.

In contrast to the experiments carried out until now, the discriminative process to evaluate this dataset has been done using a linear Support Vector Machine (SVM). Therefore, once the GRBM is trained, we obtain a new representation of each sample given by its hidden output activations, that is the output of the sigmoid function without binarization. This new feature vector (and its label) us used as a input to feed the SVM. This methodology has been previously used in the literature [Coates et al., 2011; Tang et al., 2012] in order to compare the discriminative capabilities of

[5]For further information about CIFAR-10, see Appendix A.4

the feature detectors learned by the unsupervised model employed. Table 3.5 shows the results for the different regularizations and the proposed MSR method. Note that we have compared the results only on clean images, since this kind of images are already quite challenging. According to the table, MSR also outperforms the

Table 3.5. Error rate (%) on the CIFAR-10 test set.

Model	**Error rate (%)**
L_2	48.86
L_1	47.02
EN	47.56
MSR	**41.36**

other regularization techniques evaluated in the CIFAR-10 database. Despite this, the result obtained is far from the state-of-the-art in this task [Goodfellow et al., 2013; Wan et al., 2013]. All of these methods use a convolutional approach that exploits the strong spatially local correlation present in natural images, assuming a specific geometry of the data. However, it is important to mention that our method does not make this assumption. The authors of [Krizhevsky, 2009] demonstrate the difficulty of learning interesting-looking feature detectors on natural images using RBMs. Actually, the best result in that work is 64.84% of accuracy on the test set, but they used an RBM with 10000 hidden units and the pre-training was done with many more images from the Tiny Images dataset. To compare our result to other methods that use the standard training set and a non-convolutional approach, the authors of [Glorot et al., 2010] used a three layer stacked auto-encoder obtaining 53.2% accuracy on the test set, slightly worse than our result with MSR.

3.6 Conclusions

This chapter has been focused on regularizations techniques for the RBM model. After a brief introduction about the theory of RBMs, the advantages of using some kind of regularization during training has been justified. The well-known L_1 and L_2 regularizations have their own advantages and limitations. The EN regularization is a way to mix both regularizations in the same framework. However, this last method applies the L_1 and the L_2 together to the same weights, which might not be the most suitable way to exploit their pros. We have presented a new regularization scheme for RBMs that applies both regularizations in a selective way on disjoint sets of weights. This new regularization method is called Mask Selective Regularization (MSR).

The MSR algorithm involves the use of some binary masks to delineate the target area of each regularization. These areas are defined using a criterion that takes into account the linear relations between the input variables. In this way, the method constrains the L_1 regularization to be applied on weak connections where we want to force real zeros. On the contrary, the L_2 regularization is applied to avoid that some weights, where the feature detectors are being created, to grow too large. This combination ensures both sparse connections to be robust against noise and generalization capabilities. Merging MSR with the CD algorithm enables to train the RBM with this regularization, which is able to dynamically adapt according to the current state of the weights. One of the most interesting topics of this method is that, the criterion to choose which connections are regularized by each method is valid for any kind of data, not only images. This is one of the main drawbacks from other related works on the same topic, which is solved by the MSR.

A set of experiments have been conducted to validate the method proposed in several discriminative tasks. We have used some datasets with different types of data like images of handwritten digits, text data with newsgroups and natural images. Besides the standard evaluation with the test clean data, we have also evaluated the performance on a noisy version of the test set to demonstrate the robustness of the feature detectors learned by MSR. The results show a clear advantage of using MSR compared with the results obtained with other regularization techniques. MSR is not only robust in presence of noise but also performs well on clean data comparing with other state-of-the-art techniques that use similar methods.

Several directions for future research remain. Even though the results of the experiments performed showed the advantages of the MSR regularization, these advantages may be diluted when applying the fine-tuning. According to the results, it is clear that the pre-trained features obtained with MSR are more robust than those obtained with other regularizations. However, it would be quite interesting to extend this kind of regularization also to the supervised optimization in the Backpropagation algorithm. This can be seen as a selective *dropout*, where the connections to remove follow a smarter criterion given by the binary masks. Another possible direction of future work is to extend the applicability of MSR to other type of unsupervised models such as Deep Boltzmann Machines or Deep Auto-Encoders, and check if the improvements discover for RBMs also apply to these models. Finally, to complete the evaluation of the method it would be worthwhile to evaluate the regularizations studied from a generative point of view, also varying other parameters related like the steps in the CD algorithm or using its persistent version (PCD).

Chapter 4

Local Deep Neural Networks

In the previous chapter, the benefits of restricting the learning of the RBM model to local regions have been demonstrated, specially dealing with images. This chapter describes a model that somehow follows a similar idea but applying it to discriminative networks.

4.1 Introduction

When constructing a classifier in the context of the Machine Learning, it is very important to have a powerful feature extraction method that learns the best representation of the data for the problem at a hand. In the case of images, this process can be performed by focusing on the entire image to extract a *single global representation*. Therefore, each image is represented by a single feature vector, and a discriminative rule is applied to classify an unseen object given a new feature vector. This kind of methods, such as the Principal Component Analysis (PCA) and the Linear Discriminant Analysis (LDA), have been used widely in the face recognition problem to obtain a new global representation of each face [Li et al., 2009; Ruprah, 2012].

However, in some tasks it is difficult to learn useful information directly from the entire image due to the high variability of the data. In this spirit, a *local-based* approach focuses on extracting useful information from specific parts of the image. Therefore, in this case each sample is represented by *several local representations*, so that there are indeed several feature vectors related to the same image. The discriminative rule is learnt from the local representation manifold, and it is applied to every feature vector separately. Actually, the feature vectors related to the same image can be classified into different classes, so a framework that takes into account the ensemble of local contributions to predict a final label for the new object is required. This kind of methods have been shown effective in the image database retrieval task [Mohr et al., 1997; Schmid and Mohr, 1997], and in the face recognition problem [Paredes et al., 2001; Villegas et al., 2008].

Focusing on neural networks and Deep Learning methods, it turns out that learning from the entire image using Deep Neural Networks (DNNs) has proven to be quite difficult in challenging tasks [Krizhevsky, 2009]. To deal with this problem, this chapter presents a novel discriminative model called Local Deep Neural Network (Local-DNN) which is based on two key concepts: local representations and deep networks. The assumption of this method is that it is possible to create a more reliable classifier by taking into account several local contributions learned with neural networks. Therefore, the Local-DNN model is a *local-based* approach where the learning is performed using a DNN on this local representation manifold. More specifically, this model is designed to deal with computer vision problems, where the input objects are images and the concept of local representations has a straightforward application.

The rest of the chapter is organized as follows:

- Section 4.2 explains the motivation behind this idea and the main contributions related.
- Section 4.3 resumes some interesting works concerning the idea of learning from local contributions, more specifically in computer vision problems.
- Section 4.4 presents and explains in depth the novel Local-DNN model.
- Section 4.5 evaluates the method presented using several image datasets and compares de results with other state-of-the-art methods.
- Section 4.6 draws some conclusions about the chapter and with regard to future work.

4.2 Motivation and Contributions

Focusing on images, the results obtained in the previous chapter showed that the RBM model obtains spatially localized features that detect patterns in different parts of the image. This method works well in simple tasks, such as the MNIST handwritten digits dataset, where the samples are quite small and the numbers displayed are well detected, cropped and aligned. However, using standard neural networks with bigger and/or challenging images is usually more difficult. Actually, the results already presented in Table 3.5 showed a poor performance with the CIFAR-10 task in comparison with other ML methods [han Lin and Kung, 2014; Snoek et al., 2012; Sohn and Lee, 2012], even using a pre-trained network. Related to this, [Krizhevsky, 2009] already proposed to learn independently from small windows extracted from the images (patches), due to the RBM's inability to extract meaningful features when all the pixels in the image are connected to all the visible neurons. According to the results obtained, the combination of these local learners behaved very well because it allowed the network to learn useful weights. In the neural networks context, this fact is mainly due to the difficulty of capturing the huge variability of the data modeling each pixel of the image with a dedicated connection. In contrast, learning from local regions might allow to be more robust to typical variations presented in images like translations, rotations, etc.

Besides the troubles explained before, there is another issue in terms of the number of connections. The layers in a Deep Neural Network (DNN) are *fully-connected*, which means that all the units in one layer are connected to all the units in the next layer. In case of high-dimensional inputs, such us an image of 200×200 pixels, 1.6×10^8 connections are necessary to connect the input layer to a hidden layer with 4,000 units. This situation makes the model hard to train, and the network would easily overfit the data due to the huge number of free parameters. One naive solution to this problem might be to work with low-resolution images by subsampling the original ones, at the expense of losing information [Nair and Hinton, 2010]. However, for some specific tasks it is necessary to use images with enough resolution, so that important discriminative information is not lost. An alternative choice is to use *locally-connected* layers, where each hidden unit is connected only to a subregion (receptive field) of the input image, so that some connections of the network specialize in local regions. This restriction greatly reduces the number of parameters and the overfitting of the model, and it has worked well in some image object recognitions tasks [Uetz and Behnke, 2009].

Convolutional Deep Neural Networks (DCNNs) aim to meet these requirements by learning feature detectors that are only focused on local regions of the image. This restriction reduces the number of connections between layers because the same feature detector (a set of weights) is replicated across the entire image, similar to a convolutional operation. This model has been already introduced in this thesis in Section 2.2.4. The ideas introduced by DCNNs have shown an excellent performance in computer vision tasks, obtaining the best results in several recent challenging competitions [Srivastava et al., 2014; Wan et al., 2013].

The contribution presented in this chapter is focused on a novel model called Local Deep Neural Network (Local-DNN). Given the obvious need of restricting the learning to local parts of the image, this model is inspired by the ideas of *local features* and *deep networks*. More specifically, a Deep Neural Network (DNN) is used to learn from local regions extracted from the input image called *local features*. The network learns to classify each feature according to the label of the sample (image) to which it belongs. Each of these local features is classified independently, and the final decision for the input sample is taken based on a simple voting scheme that takes into account all these local contributions. This models uses a probabilistic framework that justifies this approach. Additionally, two optional modifications of the baseline model are presented. All of these assumptions are confirmed by a set of experiments on two well-known image datasets. Besides the experiments performed in this chapter, the Local-DNN model is also validated on the gender recognition problem of face images (see Chapter 5).

4.3 State of the Art

There are several works in the DL literature that refer to the advantages of restricting the learning to small zones of the image (patches). The work in [Coates et al., 2011] gathers the most important parameters that might be important to learn useful feature detectors using unsupervised learning algorithms. If these parameters are selected correctly, it is possible to learn a powerful feature extractor model that maps small squared regions extracted from the input images to better representations of the data. Besides the carefully choice of the parameters, the only difference with the results obtained using the method followed by [Krizhevsky, 2009] (which show the difficulties of learning from natural images using RBMs) is the use of patches instead of the entire image directly. Actually, many other articles use these patches as a baseline representation to evaluate different feature learning methods [Chandra et al., 2013; Coates et al., 2012].

Using this explicit patch-based representation is less common in supervised networks [Gülçehre and Bengio, 2013; Rowley et al., 1996]. In this case, the spatial structure of the images is usually exploited by using local shared connections, as it is done in DCNNs. The first few layers of these networks are composed of convolutional filters (feature detectors), which are learned from localized regions in the input data. Furthermore, the weights learned are *tight*, which means that the same filter is reused in several regions of the image. The local learning introduced by this kind of layers have shown the ability of extracting useful representations that are usually fed to some fully-connected layers at the end of the network. DCNNs are widely used these days because their great performance dealing with images in complex task [Srivastava et al., 2014; Wan et al., 2013].

Within the framework of DCNNs, it is worth highlighting the *Network In Network* (NIN) model [Lin et al., 2013]. The NIN model proposes to replace the linear model given by the convolutional filter with a more powerful nonlinear function approximator based on micro neural network. This basic idea is similar to our Local-DNN in the way that it models different portions of the image using neural networks that share the weights across different positions. However, the entire NIN model is quite more complex and hard to train because integrates several neural networks into a generic DCNN structure.

A different approach used in the literature that employs a local-based learning method is the *locally connected* networks [Gregor and LeCun, 2010; Ngiam et al., 2010]. Unlike the DCNN model, the weights of this network are not tight, so that the filters learned in different receptive fields are not constrained to be identical. This approach raises the learning capability of the model at the expense of increasing the number of parameters. This type of local connectivity has shown to be useful in the upper layers of some DCNN models [Taigman et al., 2014], before some fully connected layers at the end.

Most of the references quoted so far refer to different ways to take advantage of local-based representations in the context of neural networks. Beyond this topic, these representations have also been used in other ML models. The work presented in [Paredes et al., 2001; Villegas et al., 2008], introduces a new approach that combines a simple local representation method with a k Nearest Neighbor classifier applied to the face recognition problem. After extracting a set of local features from the images, the final classification is performed using a probabilistic framework that combines the contribution of each local representation. The results of the experiments showed a robust performance in a face verification problem even on challenging situations such as partially occluded images. The Local-DNN presented in this chapter follows a very similar point of view. However, in this case Local-DNN takes advantage of the capability of the neural networks to learn complex distributions and to scale well to large amounts of data.

4.4 Local Deep Neural Networks

4.4.1 Introduction

This section describes the details of all the parts that build the Local-DNN model presented. First of all, a formal framework that justifies the local-based approach from a probabilistic point of view is introduced. This framework is general, but here it is particularized for the model presented. After that, it is explained how can we use a discriminative DNN to learn a local posterior probability for each local contribution. Furthermore, the possible methods to select and extract the local features from the input images are summarized. Finally, two optional modifications of the base model that employ the *location information* are explained.

4.4.2 Formal framework for local-based classification

This section addresses a formal framework to model the local feature-based classification from a probabilistic point of view [Villegas et al., 2008]. This framework is general, but we have particularized it for our Local-DNN model to deal with image data. In this case, the local features become simple windows extracted from the image at different locations (patches). Therefore, the terms *patch* and *local feature* will be used interchangeably from here onwards in this framework.

In a classification task, we denote the class variable by $c = 1, \ldots, C$ and the input pattern (image) by $\mathbf{x}$. The local features are extracted from the input pattern using some selection criterion which will be defined later. Let F denote the number of local features drawn from the input pattern $\mathbf{x}$. It is assumed that each local feature i, $i = 1, \ldots, F$, contains incomplete yet relevant information about the true class label of $\mathbf{x}$, and thus it makes sense to define a local class variable for it, $c_i \in \{1, \ldots, C\}$.

In accordance with the above idea, the posterior probability for $\mathbf{x}$ to belong to the class c is computed from a complete model including all the local feature labels,

$$p(c \mid \mathbf{x}) = \sum_{c_1=1}^{C} \cdots \sum_{c_F=1}^{C} p(c, c_1, \ldots, c_F \mid \mathbf{x}) \ , \tag{4.1}$$

which is broken into two sub-models, the first one to predict local class posteriors (from $\mathbf{x}$ only) and then another to compute the global class posterior from them (and $\mathbf{x}$),

$$p(c, c_1, \ldots, c_F \mid \mathbf{x}) = p(c_1, \ldots, c_F \mid \mathbf{x})\, p(c \mid \mathbf{x}, c_1, \ldots, c_F) \ . \tag{4.2}$$

In order to develop a practical model for $p(c \mid \mathbf{x})$, the first sub-model is simplified by assuming independence of local labels conditional to $\mathbf{x}$; that is, by application of a *naive Bayes* decomposition to it,

$$p(c_1, \ldots, c_F \mid \mathbf{x}) := \prod_{i=1}^{F} p(c_i \mid \mathbf{x}^{[i]}) \ , \tag{4.3}$$

where $\mathbf{x}^{[i]}$ denotes the part of $\mathbf{x}$ relevant to predict c_i, i.e. the *i-th* image patch.

The above approximation leads to a very simplified model. However, this simplification is based on a strong assumption of local features independence. A variation of the model that takes into account the global relationships between local class variables, for instance using Conditional Random Fields [Farabet et al., 2013], would be worth exploring. Similarly, the second sub-model is simplified by assuming that the global label only depends on local labels,

$$p(c \mid \mathbf{x}, c_1, \dots, c_F) := p(c \mid c_1, \dots, c_F) \ . \tag{4.4}$$

The above simplifications are clearly unrealistic, though they may be reasonable if each local feature can be reliably classified independently of each other. In such a case, we may further simplify the second sub-model by letting each local feature i contributes for c_i in accordance with a predefined *reliability weight*, i.e. α_i,

$$p(c \mid c_1, \dots, c_F) := \sum_{i=1}^{F} \alpha_i \, \delta(c_i, c) \ , \tag{4.5}$$

where $\delta(\cdot, \cdot)$ is the *Kronecker delta* function; $\delta(c_i, c) = 1$ if $c_i = c$; zero otherwise. In addition, $0 \leq \alpha_i \leq 1$, $i = 1, \dots, F$, and $\sum_i \alpha_i = 1$. In the simplest case, we may consider all the local features equally reliable,

$$\alpha_1 := \alpha_2 := \cdots \alpha_F := \frac{1}{F} \ , \tag{4.6}$$

but in general, α_i should be related to the discriminative power of each local contribution, or at least some subrogate measure. The model leaves open the possibility of using this parameter, so several strategies might be evaluated.

Substituting Eqs. (4.3) and (4.5) into Eq. (4.2), and then going back to our starting model (see Eq. (4.1)), we may rewrite it as,

$$p(c \mid x) := \sum_{c_1=1}^{C} \cdots \sum_{c_F=1}^{C} \left(\prod_{j=1}^{F} p(c_j \mid x^{[j]}) \right) \sum_{i=1}^{F} \alpha_i \, \delta(c_i, c) \tag{4.7}$$

$$= \sum_{i=1}^{F} \alpha_i \sum_{c_1=1}^{C} \sum_{c_2=1}^{C} \cdots \sum_{c_F=1}^{C} \left(\prod_{j=1}^{F} p(c_j \mid x^{[j]}) \right) \delta(c_i, c) \tag{4.8}$$

$$= \sum_{i=1}^{F} \alpha_i \sum_{c_i=1}^{C} p(c_i|x^{[i]}) \, \delta(c_i, c) \underbrace{\sum_{c_1=1}^{C} \cdots \sum_{c_{i-1}=1}^{C} \sum_{c_{i+1}=1}^{C} \cdots \sum_{c_F=1}^{C} \prod_{\substack{j=1 \\ j\neq i}}^{F} p(c_j|x^{[j]})}_{=1} \tag{4.9}$$

$$= \sum_{i=1}^{F} \alpha_i \sum_{c_i=1}^{C} p(c_i \mid x^{[i]}) \, \delta(c_i, c) \tag{4.10}$$

$$= \sum_{i=1}^{F} \alpha_i \, p(c \mid x^{[i]}) \ , \tag{4.11}$$

which is simply a weighted average over all local class c posteriors.

Finally, the global decision that classifies the input sample is defined by a Bayes decision rule given by choosing a class with maximum weighted sum of local posteriors,

$$\mathbf{x} \rightarrow c(\mathbf{x}) = \underset{c}{argmax}\; p(c|\mathbf{x}) = \underset{c}{argmax} \sum_{i=1}^{F} \alpha_i\, p_c^{[i]} \;, \tag{4.12}$$

where $p_c^{[i]} = p(c \mid \mathbf{x}^{[i]})$. This decision is made during testing, and aims to take into account all the local contributions to make a robust decision on the final label assigned to the input image.

An alternative to perform the final classification is that each local feature choses a class according to its maximum local posterior, so each contribution casts a vote according to its most probable class. After that, the most voted class among all the local features belonging to the test image is selected as the final decision according to the expression given by

$$\mathbf{x} \rightarrow c(\mathbf{x}) = \underset{c}{argmax} \sum_{i=1}^{F} \delta(c, \underset{c'}{argmax}\; p_{c'}^{[i]}) \;, \tag{4.13}$$

where δ is the is the *Kronecker delta* . In this voting method, α_i is not considered, so all the votes are equally reliable.

4.4.3 A local class-posterior estimator using a DNN

The probabilistic framework explained before is based on the estimation of the local class posterior probability assigned to each local feature, i.e. $p_c^{[i]}$. This probability might be estimated using a simple non-parametric algorithm such as the k Nearest Neighbor classifier, which needs to perform a search over all the local features obtained from the training images [Villegas et al., 2008]. However, this approach does not scale well when the data grows and it is necessary to use dimensionality reduction techniques (PCA) and a fast search algorithm (*kd-tree*) to reduce the computational cost.

In contrast, the Local-DNN model employs a Deep Neural Network (DNN) to approximate the class posterior probability of each local feature, which leads to the *Local-DNN* name for our proposed approach. DNNs might take advantages of the problem at a hand because they work well with large amounts of data (the total number of patches might be huge). They are also able to learn a probability distribution over the classes directly from complex raw data using several layers of representations. This probability distribution is encoded in the output layer of the network by using a softmax function according to Eq. (2.6). If the network is trained with patches, these output values are indeed the $p_c^{[i]}$ probability that must be estimated in order to perform the final classification according to Eqs. 4.12 and 4.13. All of these ideas related to our Local-DNN model are depicted in Fig. 4.1.

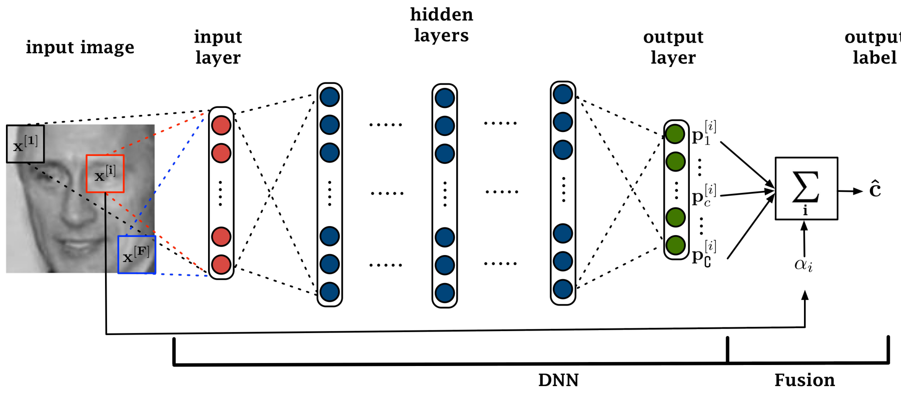


Figure 4.1. Graphical depiction of the Local-DNN model. Several patches are extracted from the input image and they are fed into a DNN which learns a probability distribution over the labels in the output layer. The final label of the image is assigned using a fusion method that takes into account all the patches' contributions.

As can be seen in the figure above, several patches are extracted from different locations in the image. These patches are fed into a DNN which has an input layer, at least one hidden layer, and an output layer to encode the classes. Once the model is trained, the DNN is able to estimate the $p_c^{[i]}$ for each patch. During testing, all the contributions learned from the patches extracted from one image are merged using a fusion technique to classify the image with a final label. This fusion method can be performed using Eqs. (4.13) or (4.12), depending on wether the voting approach or the summing posteriors method is used, respectively. Despite each patch may contribute poorly to the final label, the ensemble of all contributions makes a robust decision. This idea is similar to the general concept introduced by the Boosting algorithm [Schapire, 1990], where a set of *weak learners* can create a single *strong learner*. Our weak learner is the contribution to the final decision of each patch, which can be only slightly correlated to the true classification. However, the strong learner is created by merging all of these local contributions, and it is able to predict a reliable class for the input image.

4.4.4 Feature selection and extraction

The definition of a local feature in images includes a huge variability of options. For instance, there are several highly specialized handcrafted features that are known to work well with images, like Gabor, LBP, SIFT, HOG, etc. However, a much more simple feature is used in this work, which consists on extracting squared windows of size $w \times w$ at different locations in the image. As we have already stated before, these local features are called *patches*.

Regarding the locations where to extract these patches, it have been explored two common ways despite many others might also work well. The simplest and most generic case is to use a fixed sampling grid for all the images. This method uses the same locations for all the images, making the process quite simple and with only one parameter to adjust. This parameter is the step size s, so all the possible patches that fall inside the image are extracted every s pixels in both directions. A graphical representation of this method is depicted in Fig. 4.2.

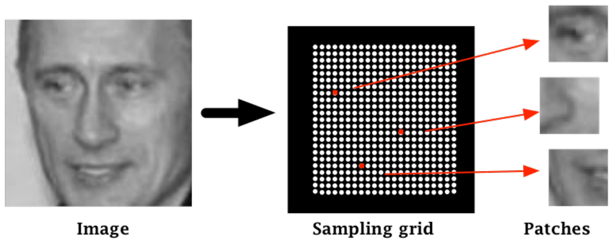


Figure 4.2. Feature extraction process using a sampling grid.

Aside from this simple method, a more smarter alternative to select these locations has been also explored. This method aims to select those patches with high information content, so that the accuracy obtained with these patches is likely to be high. Although there exist several methods to that end ([Deriche and Giraudon, 1993]), most of them are specifically designed for textured areas. Therefore, a simple and fast method that obtains a binary mask with the locations with high variability in the image is proposed. First of all, another image that emphasizes edges and translations is created using a Sobel filter. After that, a low-pass filter is applied over this image to blur these edges. Finally, the values obtained are binarized using a threshold to obtain the binary mask. A graphical representation of this method is depicted in Fig. 4.3.

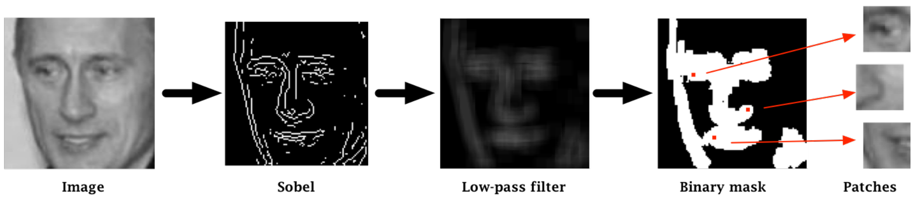


Figure 4.3. Feature extraction method that looks for high information content areas.

It is important to note that this method is less generic than using a sampling grid, and it make sense only for specific tasks where some prior knowledge can be applied to select the most informative parts of the image (a kind of *saliency map*). Actually, it has been only used in the experiments performed on the gender recognition problem using faces images, summarized in Chapter 5. In this specific task, it does makes sense to discard those patches which associated probability of accuracy is likely to be very low. For instance, uniform patches without any texture might be useless to classify the gender of the person, and it will contribute poorly to the final decision.

A similar method using face images has been used in [Paredes et al., 2001] with good results.

4.4.5 Location information and reliability weight

Section 4.4.2 presented the theoretical framework on which the Local-DNN model is based. This section proposes two modifications that increase the performance of the model in some cases. These improvements are described bellow, and both of them have been evaluated in the experiments section.

The simplifications performed in the probabilistic framework to obtain the expression (4.11) are based on the strong assumption that the placement of each patch is not relevant for the final decision. This topological information might be useful to enhance the contribution of each local feature, as it happens in convolutional neural networks. To evaluate this phenomenon we propose a slight modification of the model. This modification makes the local feature $\mathbf{x}^{[i]}$ to encode not only the image patch itself, but also the location where it was extracted. In the context of DNNs, this information is introduced in the input layer of the network by adding some extra neurons. Therefore, this layer is modified to be composed with as many units as pixels are in the patch image, plus two extra units that encode the horizontal and vertical image coordinates of the center of the patch. Our assumption is that this modification allows the network to use the location information together with the content of the patch to learn a more accurate discriminative function over the patches. It is important to stress that this improvement must be added during both the training and testing steps, so that the patches extracted in both cases must include the location information.

The second modification is related to the use of different reliable weights α_i during testing. Using Eq. 4.12 to obtain the final decision allows to assign different weights to each local contribution. In the simplest case, all the features selected by the binary mask are equally reliable when the final label is defined, with $\alpha_i = 1/F$, being F the number of local features considered in each image. The slight modification proposed is to relate the weight α_i to the accuracy of the local classifier at each position. This means that higher α_i values correspond to local features with high probability of accuracy, and vice-versa. This accuracy is obtained as an empirical estimation by considering only the placement of the patch but discarding its content. Therefore the weight associated to a specific location only depends on the mean classification accuracy of all the patches belonging to that location. These weights are estimated using the training dataset once the model has been trained. It is important to make it clear that this modification is independent from the previous one explained, despite both of them use the location of the patch.

4.5 Experiments

Once all the details about our Local-DNN model have been described, this section focuses on the experiments carried out and the results obtained regarding this model. First of all, a common protocol followed in all the experiments is described, which defines the training and testing steps, the parameters of the network, etc. After that, we focus on the results obtained on the CIFAR-10 dataset, which represents a challenging task composed of natural images. Finally, the results obtained with the popular MNIST handwritten digits dataset are also presented. It is important to note that, besides these databases, the Local-DNN model has been also evaluated in the gender recognition problem using face images. These last results are available in Chapter 5.3.

4.5.1 General protocol

First of all, this section describes the training step of the Local-DNN model. This step involves the training of the Deep Neural Network (DNN) model using the training set of the current dataset evaluated. In this way, most of the training samples are used to learn the parameters of the network, whereas the remaining ones are employed for the validation process. Once these subsets are defined, the local features (patches) are extracted from all the images, so the subsets of training and validation are comprised of all the patches belonging to the subsets of training and validation images, respectively. After that, the DNN is fed with these patches and their labels, and the network is trained using the standard Backpropagation algorithm. Basically, this algorithm modifies the weighted connections between the layers according to the gradient of the error calculated between the real labels and the predicted ones (see Section 2.2.3). The training process is performed for a limited number of epochs, and the network with the lowest classification error on the validation subset is selected.

Once the training process of the DNN is done, the model can be evaluated using the testing data of the current dataset. According to the definition of the Local-DNN model, the DNN is able to obtain a probability distribution on the classes for each patch. However, to evaluate an input image it is necessary to classify it among all the possible labels. Therefore, the local results obtained by the DNN must be merged to predict a single class for the input image. In this way, for each test image all the patches are extracted using the same method employed during training. Each of these patches is forwarded through the network to obtained a probability distribution for each of them. Finally, a final label for the test image is predicted taking into account the contributions of all the local features of the image using a fusion method. Let us remember that this step can be done summing the probability distributions (see Eq. (4.12)), or using a direct voting scheme (see Eq. (4.13)). The results obtained for all the test images in the current dataset are averaged.

Regarding the configuration of the network, the number of hidden layers in the experiments is changed in order to evaluate the improvement obtained by using deeper architectures. All of these layers are composed with 512 Rectified Linear Units (ReLU). This type of units (see Section 2.2.2) have shown to learn much faster in networks with many layers, instead of the traditional sigmoid ones [Glorot et al., 2010]. On the other hand, it is obvious that the size of the input and the output layers depend on the problem at hand.

4.5.2 Experiments with CIFAR-10

The first experiments to evaluate the Local-DNN model are performed using the CIFAR-10 dataset (see Appendix A.4). The images of this dataset are RGB images of size 32×32 pixels with values in the range $[0, 255]$. First of all, the data is rescaled so that all the pixel values lie in the range $[0, 1]$. After that, the local features are extracted from the images using the simplest method proposed (see Fig. 4.2). All the windows of size $w \times w$ are extracted in the positions given by a dense, equispaced sampling grid with a unit sample spacing ($s = 1$). Note that the patches with a region that lie outside the image are discarded. Therefore, each patch can be represented as a feature vector of pixel intensity values in $\mathbb{R}^N$, with $N = w \cdot w \cdot d$ and $d = 3$ being the number of image channels. After that, every patch is normalized to zero-mean and unit-variance. This is done for all the images in each subset of the database. Note that the dimensionality of these feature vectors can also be larger ($\mathbb{R}^{N+2}$) if the location information of the patch is included according to the first modification explained in Section 4.4.5. Table 4.1 summarizes the number of images and patches obtained after this extraction method for the different patch sizes evaluated (w). Note that the *train* and *valid* subsets together form the $50K$ images of the original CIFAR-10 training set. According to the table, it is clear that there is a tradeoff between the patch size parameter and the number of patches extracted, so that the larger the size of the patch, the fewer patches will be valid because more of them would fall outside the image.

Table 4.1. Number of images and approximate number of patches extracted from each subset in the CIFAR-10 dataset depending on the patch size. Note that $1K$ = one thousand and $1M$ = one million.

Subset	**Num. Images**	**Approx. Num. Patches**		
		Path Size		
		5×5	**10×10**	**15×15**
train	45K	30.5M	21.8M	11.5M
valid	5K	3.4M	2.4M	1.3M
test	10K	6.8M	4.8M	2.6M

Once the patches are extracted, the Local-DNN model is trained and evaluated following the procedure explained in Section 4.5.1. All the results obtained for this dataset are summarized in Table 4.2. As it can be seen in the table, for each patch size evaluated, the number of hidden layers is also changed. Note that the last row results of each block refers to the experiments performed including the location information where the patch was extracted in the local feature. This modification has been tested only for the best case regarding the number of hidden layers. Regarding the results obtained, there are two types of results displayed. On the one hand, we can see the accuracy obtained by the network at a patch level (**Patch Acc.**). With these results we can get an idea of how well the network is able to learn a mapping between the patches manifold to the classes. On the other hand, the test accuracy obtained at image level is also displayed (**Image Acc.**) using the two fusion methods

Table 4.2. Classification accuracy at image and patch level on the test set for the CIFAR-10 dataset, for different configurations of the Local-DNN model.

Patch Size	**Num. Hidden Layers**	**Patch Acc. (%)**	**Image Acc (%)**		
			Voting	**$\sum$ Post.**	
				$\alpha = 1/F$	**$\alpha \propto$ acc**
5 × 5	1	30.57	49.88	49.88	50.34
	2	35.83	61.57	63.68	64.09
	3	37.17	64.92	67.03	67.39
	4	36.86	66.40	68.15	68.49
	4 + loc	41.14	71.94	74.04	74.16
10 × 10	1	40.01	60.13	60.34	60.51
	2	49.06	74.52	75.19	75.13
	3	49.31	77.81	78.61	78.60
	4	49.15	75.64	76.39	76.21
	3 + loc	53.44	80.41	**80.73**	80.54
15 × 15	1	48.90	64.24	64.79	64.73
	2	56.85	76.52	76.84	76.82
	3	56.34	76.08	76.59	76.55
	3 + loc	**59.57**	76.96	77.23	77.25
32 × 32	1	51.29			
	2	52.11			
	3	51.89			

evaluated: the voting scheme or summing the posterior probabilities of each local contribution. In this latter case, it has been made a distinction between the case where all the features are equally reliable ($\boldsymbol{\alpha}= \frac{\mathbf{1}}{\mathbf{F}}$), and the case where each local feature is weighted proportionally to its location patch accuracy ($\boldsymbol{\alpha} \propto \mathbf{acc}$). Finally, it is important to mention that the results obtained using the entire image as a patch are also included ($w = 32$) at the bottom of the table. In this case there is only one accuracy result given by the neural network without any fusion step because there is only one patch per image, i.e. the image itself. This result obtained with a standard DNN is used as a baseline to evaluate the improvement obtained with the Local-DNN model.

Once the structure of the table is understood, several conclusions can be drawn according to the results obtained depending on the parameter evaluated:

- **Patch size**: there is a clear gap between the patches of size 5×5 and the others. Probably, a window of 5×5 pixels is not informative enough for the network despite the largest number of patches extracted according to Table 4.1. Using patches of 10×10 pixels yields the best results, which are slightly worse than those obtained with the size of 15×15. Comparing these values with our baseline result, it is clear the difficulty of learning directly from the entire image using the image itself as a patch (32×32 pixels).

- **# hidden layers**: it can be seen that there is a big gap between the network with one hidden layer and the networks with two or more hidden layers. This issue occurs at patch and image level, and denotes the poorer representation power of a shallow network with just one hidden layer in this case. According to the results, a 3-layer network is deep enough to learn the complex mapping between the manifold represented by the patches and their classes. It is also interesting the lack of improvement obtained using more hidden layers when the entire image image is employed. In this case, the network is not able to take advantage of a deeper architecture.

- **Location information**: there is a clear improvement obtained by including the location information in the local feature for all the patch sizes evaluated. This improvement is quite clear at a patch level, which indicates that this information is relevant for the network in order to learn to classify a patch, which is natural. For instance, if we know that a uniform blue patch is located at the top of the image, it makes sense to think that there is a sky in that image. Following with this conclusion, it can also be appreciated that this improvement is maintained at a patch level, but decreases at image level when increasing the size of the patch. This probably happens because big patches contain more representative information content, and the location information contributes relatively little in the final decision. Note that there is only one location when using the entire image as a patch and there is no sense to include this information in this case because it is always the same.

- **Fusion method**: focusing on the testing parameters, summing the posterior probabilities of the local patches ($\sum$ **Post.**) yields slightly better results than if each patch votes the class according to its maximum local posterior (**Voting**). This result might be explained because, in the former case, the uncertainty of a patch among the classes is taken into account, while in the latter case it is not because the model only uses the maximum posterior of each patch to emit a vote despite this probability might be low. However, there are no big differences between both methods.

- **Reliability weight**: this modification enables the use of a non-uniform weighted sum of contributions ($\boldsymbol{\alpha} \propto \mathbf{acc}$) when summing posteriors, as explained in Section 4.4.5. However, the results obtained are very similar to those obtained using equally reliable local features ($\boldsymbol{\alpha} = \frac{\mathbf{1}}{\mathbf{F}}$), and there is not significant improvement. This is probably because the accuracy of the patches regarding its location is almost uniform in this specific task. To illustrate this issue, Fig. 4.4 displays the probability of accuracy of the DNN depending on the position where the patch was extracted, for the best configuration in the table. According to the image, the *best* patches are extracted around the center of the image, which is natural because the main object of each image is displayed centered. However, it is important to note that the lowest probability (dark color) is around 0.40, and the highest probability (white color) is around 0.59. The difference between these values is quite small, hence the use of different weights might be pointless in this case because the same result is obtained if all the patches are equally weighted.

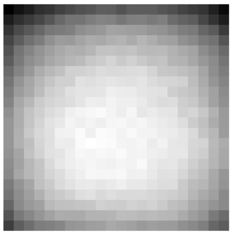


Figure 4.4. Probability of accuracy of the DNN at a patch level depending on the position where the patch was extracted for the CIFAR-10 dataset. Light and dark colors denote high and low probability, respectively. Note that the lowest probability is around 0.40 and the highest probability is around 0.59.

Summing up all these conclusions, the network that obtains the best performance has 3 layers depth, uses patches of size 10×10 including the location information in the local feature, and computes the final decision using the summing posteriors method with equally reliable weights. The result obtained is 80.73% of accuracy on the 10000 CIFAR-10 test images, which improves greatly the accuracy obtained feeding the entire image to the network, as can be seen at the bottom of Table 4.2. Furthermore, Table 4.3 compares our best result with other results obtained using different methods published in the literature with the same dataset. According to this table, it is clear that our Local-DNN is far from the best state-of-the-art results obtained in

Table 4.3. Classification accuracy on the test set of the CIFAR-10 dataset for different methods.

Method	Accuracy (%)
Stacked Den. Autoencoders [Glorot et al., 2010]	50.48
RBM + fine-tuning [Krizhevsky, 2009]	64.84
Improved LCC [Yu and Zhang, 2010]	74.50
KDES-A [Bo et al., 2010]	76.00
PCANet-2 [Chan et al., 2014]	78.67
K-means Tri. [Coates et al., 2011]	79.60
ConvNet + Dropout [Hinton et al., 2012]	84.40
Multi Column DNN [Schmidhuber, 2012]	88.79
Maxout Net [Goodfellow et al., 2013]	90.65
DropConnect [Wan et al., 2013]	90.68
Network in Network [Lin et al., 2013]	91.20
Deeply Supervised Net [Lee et al., 2015]	91.78
Local-DNN	80.73

this database. However, most of these methods are more complex because they use large convolutional neural networks that demand high computational resources and large datasets. In contrast, our method is quite simple in both testing and training steps, and it would work well in tasks with very few samples because it implies a very large increase of the number of available training samples (note that from the original image we can obtain thousands of smaller patches). For these reasons, the results obtained are still encouraging compared to other simple methods that obtain worse results. It is also interesting to analyze the first two values of the table. These results are obtained using non-convolutional networks and learning directly from the entire image. Like our baseline result, these poor performance confirms the advantage of learning from small regions, specially with complex data as it happens in this case.

4.5.3 Experiments with MNIST

The following experiments evaluate our Local-DNN model using the MNIST handwritten digits dataset (see Appendix A.1). The original images of this dataset are binary images of size 28×28 pixels, even though they contain some grey levels as a result of the anti-aliasing technique employed by the normalization algorithm. From these images, all the windows of size $w \times w$ are extracted in the positions given by a sampling grid with $s = 1$. This sampling grid is defined as a 20×20 pixel box centered on the image because this size should let the patches to sweep the most important areas of each image. This specific size was selected because all the digits in the database fit into a box with this scale after the normalization process. In contrast to the feature extraction process in the CIFAR-10 dataset, in this case we did not remove the patches with a region that fall outside the image. Instead we expanded the original black background of the original images to keep all the possible patches.

For this dataset, each patch can be represented as a feature vector in $\mathbb{R}^N$ of pixel values, with $N = w \cdot w \cdot$ because there is only one channel. Note that these feature vectors can also be in $\mathbb{R}^{N+2}$ when the location information is included in the local feature.

Table 4.4 summarizes the number of images and patches obtained after the extraction process. Note that the same number of patches is obtained regardless the different sizes evaluated. For instance, for the *test* subset there is a total of $4M$ patches, obtained by multiplying the size of the grid times the number of images, i.e. $20 \times 20 \times 10000$. Note also that the *train* and *valid* subsets together form the $60K$ images of the original MNIST training set.

Table 4.4. Number of images and number of patches extracted from each subset in the MNIST dataset.

Subset	Num. Images	Num. Patches
train	54K	21.6M
valid	6K	2.4M
test	10K	4M

Once all the patches are extracted we train and evaluate our Local-DNN model following the procedure explained in Section 4.5.1. The results obtained are summarized in Table 4.5, which is similar to that used with the CIFAR-10 dataset in Section 4.5.2. Note that, unlike those results, this table shows the error rate instead of the accuracy just because it is common in the literature to use this measure with the MNIST dataset.

To evaluate the results displayed in the table, a baseline result is used to evaluate the improvement obtained. This result is 1.2% of error rate on the test set, and it was obtained by G. Hinton with a standard neural network with several pre-trained layers in his famous paper [Hinton and Salakhutdinov, 2006]. First of all, let us forget the results in the last row of the table and focus on those where the patch size is smaller than the image size to draw some conclusions from different points of view:

- **Patch size**: it is clear that using small patches (10×10) does not work well in this problem. The patch size must be increased to, at least, 15×15 pixels to greatly improve the baseline result. This phenomenon is probably due to the difficulty of learning from small parts of the digits that are not representative enough for the network. Actually, the best results are obtained with patches that are almost as big as the digits themselves, which leads away somewhat the basic idea of our model of *learning from local regions* in the image.

- **# hidden layers**: the difference at the error rate level between a network with one hidden layer and networks with two or more hidden layers is also clear in these experiments. Again, the discriminative model learnt by the shallow network is far less robust than the deeper model in all cases.

[1]This experiment was performed using a larger sampling grid of size 28×28 (the entire image), in order to obtained the largest number of patches available for each image.

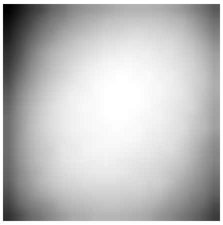
Table 4.5. Classification error rate at image and patch level on the test set for the MNIST dataset, for different configurations of the Local-DNN model.

Patch Size	**Num. Hidden Layers**	**Patch Error rate (%)**	**Image Error rate (%)**		
			Voting	**$\sum$ Post.**	
				$\alpha = 1/F$	**$\alpha \propto$ acc**
10×10	1	47.67	9.32	4.94	4.75
	2	40.93	4.24	2.22	2.18
	3	40.12	4.40	2.01	1.89
	3 + loc	23.33	1.44	1.06	1.04
15×15	1	23.90	2.02	1.74	1.71
	2	18.57	1.05	0.84	0.86
	3	18.30	0.87	0.69	0.68
	3 + loc	9.59	0.61	0.58	0.61
20×20	1	10.09	1.16	1.12	1.14
	2	7.14	0.55	0.51	0.51
	3	7.06	0.56	0.55	0.56
	3 + loc	3.65	0.55	0.55	0.55
25×25	1	4.47	0.94	0.96	0.94
	2	2.96	0.53	0.54	0.55
	3	2.91	0.54	0.54	0.54
	3 + loc	**1.81**	0.58	0.58	0.58
28×28[1]	3	5.22	**0.47**	**0.47**	0.48

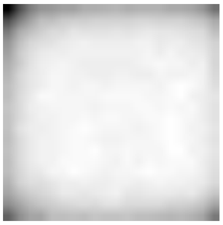
- **Location information**: according to the results, the location information using small patches improves greatly the performance at both error levels (patch and image). However, this improvement vanishes at image level as the size of the patch is increased. Curiously, using this information with big patches helps the network to better classify the patches (the error at patch level is quite lower), but this progress does not lead to an improvement once the fusion method is applied.

- **Fusion method**: a similar phenomenon occurs regarding this parameter, since the voting method yields worse results than summing the posteriors when the patch size is 10×10 or 15×15. However, both methods behave very similar with larger sizes.

- **Reliability weight**: this modification introduced during testing does not improve the result obtained by the fusion method. To deeply analyze this result, Fig. 4.5 displays the probability of accuracy of the DNN depending on the position where the patch was extracted, as we did with the CIFAR-10 dataset. A pixel in that image indicates how likely is that the patch extracted in that position predicts the correct class. The left subfigure is the result for the network trained with patches of size 10×10, while the right one is the result for the network trained with patches of size 25×25. In both cases the best patches are centered. However, there is a great difference between the lowest and the highest probability in both cases (see the caption for the specific values). According to these results and those displayed in the table, it is quite surprising that using different weights in the case of 10×10 patches does not improve the results significantly. Therefore, this means that taking into account with the same weight the patches which result is probably to be inaccurate does not worsen the final result obtained.



(a) Network trained with patches of size 10×10. Note that the lowest probability is around 0.25 and the highest probability is around 0.95.



(b) Network trained with patches of size 25×25. Note that the lowest probability is around 0.92 and the highest probability is around 0.99.

Figure 4.5. Probability of accuracy of the DNN at a patch level depending on the position where the patch was extracted for the MNIST dataset. Light and dark colors denote high and low probability, respectively.

The analysis of the results reveals that our Local-DNN model does not take advantage of learning from small regions in this specific task. Therefore, the results obtained encouraged us to use bigger patches to obtain the highest performance in this dataset, despite the fact that the main idea of *local features* recedes into the background. The results displayed in the last row of the table are obtained with an extra experiment that uses patches with size 28×28 pixels (the same size as the original images), and enlarges the sampling grid to be as big as the entire image. According to these values, $28 \times 28 = 784$ different patches centered in all the possible pixels of each image are extracted. Using a 3-layer neural network with any of the fusion methods available, 0.47% error rate is obtained on the MNIST test set, i.e. 47 errors out of 10000 images. This great result is far better than the baseline of 1.2% obtained with a standard neural network. However, it is important to say that, in this database, this improvement is indeed given by using several translated versions of the original images since the *local features* are actually the same size of the original images. We have also evaluated this network trained with several patches, using only the test images without extracting patches during testing (without any fusion method). The result obtained is 1.33% error rate on the test set, which means that it is important to use several patches during both training and testing. Therefore, the fusion method performed by the Local-DNN model allows to take advantage of learning from several contributions.

It is also interesting to analyze the higher error rate at a patch-level obtained in this experiment (5.22%) in comparison with the rest of the experiments. This result is because there are more patches centered near the edges of the images in this case, due to the bigger sampling grid employed. These patches belong to meaningless portions of the digit, so it is more difficult to classify them and the error rate increases. However, the result at image level including these, a priori, poorer contributions is improved. It is therefore concluded that using as many as possible patches is quite important as long as these patches contribute slightly better than random to the final decision.

Finally, it is interesting to compare the performance obtained with other state-of-the-art results in the same dataset, as we did with the CIFAR-10 database These results are summarized in Table 4.6 According to the results, our Local-DNN obtains

Table 4.6. Error rate on the test set of the MNIST dataset for different methods.

Method	**Error rate (%)**
3 layer NN + pretrain [Hinton and Salakhutdinov, 2006]	1.20
Deep Boltzmann Machines [Salakhutdinov and Hinton, 2009b]	0.95
PCANet-2 [Chan et al., 2014]	0.62
Invariant SVM [Decoste and Schölkopf, 2002]	0.56
Network in Network [Lin et al., 2013]	0.47
Maxout Net [Goodfellow et al., 2013]	0.45
Deeply Supervised Net [Lee et al., 2015]	0.39
MLP + elastic dist. [Ciresan et al., 2010]	0.35
Multi Column DNN [Schmidhuber, 2012]	0.23
DropConnect [Wan et al., 2013]	0.21
Local-DNN	0.47

a very competitive performance compared to other state-of-the-art methods. Our method outperforms other non-convolutional networks [Hinton and Salakhutdinov, 2006; Salakhutdinov and Hinton, 2009b] and simple methods [Chan et al., 2014; Decoste and Schölkopf, 2002]. However, analyzing the results in depth, the importance of using data augmentation techniques with this kind of data (handwritten digits) draws special attention. Most of the best results are obtained using these techniques to obtained translated, flipped, rotated and scaled versions of the original images. Actually, [Ciresan et al., 2010] demonstrated that it is possible to obtained 0.35% error rate on the MNIST test set by just using a large standard neural network fed with well designed elastic distortion versions of the digits. This result along with ours reveals that standard DNNs work well learning from the entire image in this case, and therefore restricting the learning to local regions has less impact in the final result in this database.

4.6 Conclusions

This chapter presents a novel discriminative model called Local Deep Neural Network (Local-DNN), which is focused on computer vision problems. This model is based on two key concepts: local features and deep architectures. On the one hand, a neural network learns to classify small patches extracted from images according to the label of the image to which it belongs. On the other hand, using these local contributions, a simple fusion scheme that merges these contributions is applied to classify the entire image within a single class. Note that this fusion can be performed using a voting method or summing the posterior probabilities given by the neural network. The idea of this local-based classification is justified through a generic framework from a probabilistic point of view. Also, two modifications of the model have been proposed to increase its performance. The first one is based on including the location information where the patch was extracted into the local feature. The second one focuses on assigning different weights to each contribution during testing, depending on the accuracy of the location where the patch was extracted.

The first set of experiments to evaluate the Local-DNN model has been performed using the CIFAR-10 dataset. The results show the difficulty of a standard neural network to learn from the entire image regardless the depth of the network. In contrast, our Local-DNN model performs much better with several layers, obtaining better results than other simple methods published. However, the 80.73% of accuracy obtained is still far from the state-of-the-art results in this tasks, which are usually based on complex convolutional approaches that usually need large datasets. Regarding the modifications introduced, it is clear that the location information where the patch was extracted must be included in the local feature to obtain the best results, specially when the size of the patch is small. In contrast, using the accuracy of each patch depending on its locations to weight each local contribution turns out not to be meaningful. The results obtained with this modification are quite similar to those obtained using equally reliable features. Therefore, all the local contributions are important to the final decision despite the fact that some of them have a lower accuracy.

Some experiments have been also performed using the MNIST handwritten digits dataset. Despite the fact that the Local-DNN model does not benefit from using small patches in this specific task, it is possible to obtain state-of-the-art results using bigger patches (several translated versions of the original images). The best result obtained is 0.47% error rate on the test set, which is similar to other results obtained using more complex models.

For future research, there are several questions that remain open. First of all, this version of the model might not take into account large variations in the scale of the images. Therefore, it might be worth considering to use several networks trained with different patch sizes. In the fusion step, all the contributions learned at different scales are merged using the same framework, so the final label is predicted using multi-scale local contributions. Another direction of research which should be followed is to consider the use of convolutional networks instead of DNNs. A DCNN has shown to be a much powerful discriminative model dealing with images, so the local contributions learned by this model might increase the performance resulting a Local-DCNN model. Finally, it would be also interesting to apply this model to other computer vision problems where the labeled data available is scarce. In this case, other neural networks might easily overfit the training data, even using regularization techniques such as the dropout. Our Local-DNN could take advantage in this type of tasks because the huge number of patches employed would allow to train the network successfully without overfitting.

Chapter 5

Application to Gender Recognition

Besides de contributions presented in the previous chapters, we have also a particular interest in biometrics applications. For this reason, this chapter aims to evaluate several Deep Learning (DL) techniques in the gender recognition problem of face images. These experiments also include the results obtained with our Local-DNN model presented in Chapter 4. According to these results, it turns out that our model outperforms other deep networks evaluated and obtains state-of-the-art results in the two datasets evaluated.

5.1 Introduction

Identifying demographic attributes of humans such as age, gender and ethnicity using Machine Learning methods has received increasing attention in recent years. Specially, gender recognition of face images is an important task in computer vision as many applications depend on the correct gender assessment. According to [Ng et al., 2012b], examples of these applications include:

- **Human-computer interaction systems.** This kind of systems can be made more human-like and they can response appropriately if they know the gender of the person to interact with.
- **Surveillance systems.** Some security systems can restrict the access to some areas depending on the gender of the person.
- **Content-based indexing and searching.** In these days there is a large amount of photos and videos being produced. It would be useful to identify the gender to improve the searching methods.
- **Biometrics.** Some applications can be trained separately based on the gender of the person to improve accuracy.

- **Targeted advertising.** Targeted advertising is used to display advertisement relevant to the person looking at the billboard based on attributes such as the gender.

Once the importance of the correct gender assessment is clear, how to perform this classification using face images can be discussed. The gender recognition problem is usually divided into several steps, similarly to other classification tasks: object detection, preprocessing, feature extraction and classification. In the detection phase, the face region is detected and cropped from the image. Then, different preprocessing techniques are used to reduce possible variations in the data such as the scale and the illumination. After this normalization, the feature extraction phase aims at obtaining representative and discriminative descriptors of the face region that are useful for the classifier. Finally, this binary classifier is trained to learn the differences between male and female, and to be able to generalize well to new samples.

Perhaps, the feature extraction is the most critical step in order to achieve good performance. The existing methods of this process can be classified into two groups: *geometric-based* and *appearance-based* methods. The former group is based on measuring distances between characteristic points in the face (fiducial points), and the later one aims to extract a new representation of the face to enhance the discriminative information and reduce the variance of the images. Traditionally, the appearance-based methods have been the most successful in the gender recognition problem thanks to the knowledge and expertise of many feature practitioners that are able to extract more robust representations than the raw pixels. Some examples of these features are Local Binary Patterns (LBP), Haar-like features, Gabor wavelets, Scale Invariant Feature Transform (SIFT) features, etc. Once the new representation of the data is extracted, a binary classifier, such as the Support Vector Machine (SVM) or the AdaBoost method, is trained to perform the classification. This generic method works well and most of the state-of-the-art results obtained in this task use this framework as a baseline [Dago-Casas et al., 2011; Shan, 2012; Tapia and Perez, 2013].

In contrast, the discriminative DL models aim to automatically discover these representations as well as to perform the classification process. Despite the fact that neural networks have been applied recently to several face recognition tasks [Huang et al., 2012a; Schroff et al., 2015; Taigman et al., 2014], there is not a specific study of DL methods applied to the gender recognition problem to our knowledge. For this reason, the experiments summarized in this chapter are aimed to obtain state-of-the-art results in this task by evaluating several models and parameters related to the DL framework. First of all, the performance of a standard neural network (DNN) has been evaluated, changing several key parameters and options such as the use of pre-training and the type of units employed. After that, a convolutional neural network (DCNN) has been also evaluated to show the strengths of this type of network dealing with images in challenging tasks. Finally, we have obtained several results with our novel Local-DNN model presented in Chapter 4 to show the great performance obtained in this specific task. All the results have been compared to other state-of-the-art techniques available in the literature.

5.2 State of the Art

As cited before, extracting a good representation of the data is perhaps the most critical step in most of the pattern recognition and machine learning problems. Initial approaches for gender recognition used the geometric relations between facial landmarks as a feature representation [Ng et al., 2012a]. However, these methods required a very accurate landmark detection and it was shown that relevant information was thrown away. For this reason, all recent approaches use *appearance-based* methods. These methods can be *holistic*, when the whole face is used to extract useful features, or *local*, when information is extracted from local regions of the face. These regions can be placed at strategic locations like eyes, nose, mouth, etc. or simply using an equally spaced grid on the image.

Regarding the type of these handcrafted features found in the literature, they can be as simple as the raw pixels [Moghaddam and Yang, 2002] or pixel differences [Baluja and Rowley, 2007] of the image. Sometimes, simple features are pooled together as in [Kumar et al., 2009], where image intensities in RGB and HSV color spaces, edge magnitudes, and gradient directions were combined. More elaborated features include Haar-like wavelets [Shakhnarovich et al., 2002], Local Binary Patterns (LBPs) [Shan, 2012] or Gabor wavelets [Leng and Wang, 2008]. These features work well and they are robust to small illumination and geometric transformations. However, they are based on the expertise of the researcher to find the best choice for a given problem.

Note that these alternative representations of the face are usually high-dimensional, and it is common to apply dimensionality reduction techniques in order to both decrease the computational requirements and extract the relevant discriminative information. These methods can be unsupervised or supervised, linear or non-linear. The two most popular algorithms are Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA), being the former unsupervised and the latter supervised. In [Villegas and Paredes, 2011], the authors show a good comparison of different dimensionality reduction methods on a gender recognition problem among others tasks. All of these techniques have been widely used because of their simplicity and effectiveness [Buchala et al., 2004; Graf and Wichmann, 2002; Turk and Pentland, 1991]. However, a possible disadvantage of this approach is that the information captured may not be relevant to represent a face in this specific problem.

Finally, regarding the binary classifier employed, different options have worked well using several representations of the data. For instance, the AdaBoost algorithm is a fast method that selects the most relevant simple features to combine them into a single strong classifier. AdaBoost have been used widely in the literature [Baluja and Rowley, 2007; Kumar et al., 2009; Shan, 2012]. Support Vector Machines (SVMs) have been used in the gender recognition problem as well [Moghaddam and Yang, 2002; Shan, 2010], using directly the raw pixel information or more complex descriptors. For instance, a recent article [Eidinger et al., 2014] use a dropout-SVM approach with LBP features to estimate both the gender and the age values in unconstrained images. In this spirit, an excellent comparison of different gender recognition methods can be found in [Dago-Casas et al., 2011].

In contrast to these approaches, DL techniques have not been applied extensively in the gender recognition problem. One of these cases is the work presented in [Bartle and Zheng, 2015], which presents a new DL model for performing the gender classification using a Natural Language Processing approach. Regarding the use of face images, [Levi and Hassner, 2015] uses a DCNN to improve the current state-of-the-art in a gender and age classification task using vey challenging images. Finally, [Jiang et al., 2014a] merged low-level and high-level features extracted using a DNN and a DCNN to improved other results obtained with LBP features and the SVM classifier.

5.3 Experiments

5.3.1 General protocol

The experiments carried out have been performed using two well-known face image datasets taken in unconstrained scenarios. These datasets are the Labelled Faces in the Wild (LFW) and the Groups/Gallagher. More details about them can be found in Appendix A.5 and A.6. The evaluation protocol for both databases splits all the images in 5 folds to perform a 5-fold cross validation process, so 4 folds are used for training and the remaining fold for testing. The experiments are performed using the 5 possible combinations and the results obtained are averaged. For the problem at a hand, the binomial confidence intervals can be calculated using a normal approximation which leads to the expression $\hat{p} \pm z\sqrt{\hat{p}\,(1-\hat{p})/n}$, where $\hat{p}$ is the mean accuracy, n is the number of test samples and the z value must be 1.96 if we want to compute the standard 95% confidence intervals. Given the large number of testing samples in these experiments, these confidence intervals would be quite small. For this reason, we have not included these intervals in the result tables presented in this section. Under this protocol, and regardless of the specific network evaluated, the models have been trained until the average cross-entropy error on the training data falls below a pre-specified threshold. To fix this threshold, the same network is trained but using only 3 folds from the training data and using the remaining one as a validation set. Then, the cross-entropy threshold value is fixed with the smallest classification error obtained on the validation set.

On the other hand, a simple preprocessing step on the images is necessary due to their unconstrained nature. This step is the same for both datasets and aims to reduce the face detection inaccuracies as far as possible. First, all the images have been aligned to a canonical pose. In the case of the LFW, a common aligned version of the database has been used [Huang et al., 2012b]. In contrast, the location information of the eyes has been used in the Gallagher's dataset to transform all the faces to a new pose with the eyes located in the same position, $(22, 27)$ and $(57, 27)$ pixels, as it was done in [Dago-Casas et al., 2011]. After that, two different face regions of the image of 105×90 and 105×105 pixels are cropped. The former region is resized to 40×32 pixels for the experiments performed with the DNNs, and the latter crop was resized to 152×152 pixels for the experiments performed with the DCNN. These sizes were selected according to previous works using the same databases [Taigman et al., 2014; Villegas and Paredes, 2011]. Moreover, using the same sizes facilitates comparison to the results obtained with the same databases. Finally, all the images are converted to grayscale, normalized to zero-mean and scaled by the average standard deviation of all the pixels in all the training images. The preprocessing step used for the Local-DNN model is slightly different. The crop faces of 105×105 pixels has been resized to 60×60 pixels due to the unmanageable number of patches obtained otherwise. The resulting images were converted to grayscale, and all the pixels values were scaled to the range $[0, 1]$.

5.3.2 Results with DNN

This section assesses the gender classification performance of a DNN with just one hidden layer, so that the network employed is not actually *deep* because it is composed of the input layer, just one hidden layer and the discriminative output layer. These experiments have been performed only for the LFW dataset, and they aim to evaluate the performance of a shallow network with respect to several parameters such as the number of hidden units, the type of these units and the use of pre-training.

In the case of using pre-training, an RBM is trained to initialize the weight connections between the visible and the hidden units of the DNN, similarly to the experiments exposed in Chapter 3. Each RBM is trained using the $CD-1$ algorithm for 100 epochs using the training set without the labels information. The weights of the RBM are initialized with small random values sampled from a normal distribution with zero mean and standard deviation of $4\sigma\sqrt{\frac{6}{n_{hid}+n_{vis}}}$, where n_{hid} is the number of hidden units, n_{vis} is the number of visible units and σ is a parameter [Bengio and Glorot, 2010]. We fixed $\sigma = 1.0$ for sigmoid units and $\sigma = 0.01$ for ReLU units. The learning rate value was set to 0.001 for both weights and biases. Some of the feature detectors learned by the RBM are represented in Fig. 5.1.

The training process is similar to that followed in the experiments presented in Section 3.5. Once the RBM is trained, the learned weights are used to initialize a DNN by adding an extra layer with two units that represent the labels of the samples (male or female). Then, the entire network is fine-tuned with the Backpropagation algorithm using the label's information. The fine-tuning process stops when the average cross-entropy error on the training data falls bellow a pre-specified threshold, as explained before. The test accuracy results for each configuration are shown in Table 5.1. According to the numbers, the ReLU units outperform the Sigmoid units even when fewer hidden neurons are employed. These ReLU units also learned faster during the pre-training and the fine-tuning process, in accordance with the results obtained by other authors [Glorot et al., 2010; Nair and Hinton, 2010]. On the other hand, there is almost no improvement when using unsupervised pre-training, probably due to the shallowness of the network. There are also no big differences varying the number of hidden units.

After evaluating the performance with one hidden layer, a second and a third layer are added to evaluate the improvement of using a deeper architecture. In these experiments, the same number of hidden units are chosen for all the hidden layers. For example, in a three-layer network, a $1280-512-512-512-2$ architecture is used, where the last layer is the discriminative layer and the input layer represents the

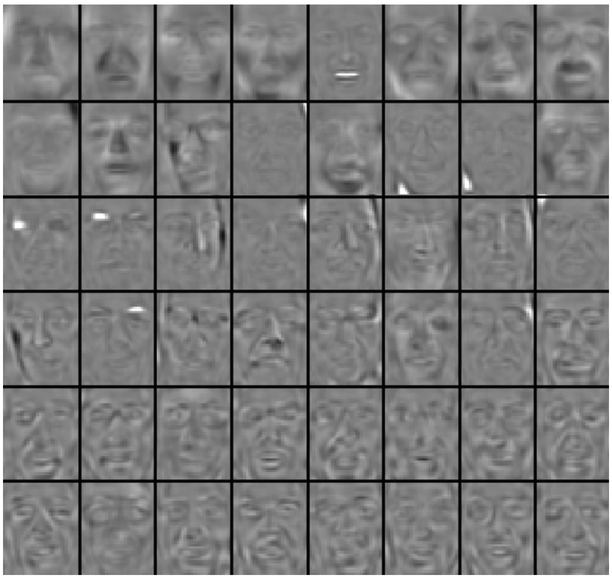


Figure 5.1. Some feature detectors learned by the RBM model.

Table 5.1. Accuracy on the test set for a one layer DNN in the LFW database.

Unit Type	**Pre-training**	**Num. Hidden Units**	**Acc. (%)**
Sigmoid	no	512	91.28
		1024	91.38
		2048	90.98
	yes	512	90.29
		1024	91.25
		2048	91.44
ReLU	no	256	91.76
		512	92.04
		1024	91.84
	yes	256	91.92
		512	**92.10**
		1024	92.09

pixels. Also, the use of the dropout technique ([Srivastava et al., 2014]) has been evaluated with the deepest network that uses ReLU units. This method randomly omits half of the outputs in every hidden layer during training, and also 20% of the units in the input layer. The results are shown in Table 5.2. As may be observed, ReLU units

Table 5.2. Accuracy on the test set for the DNN model with two and three layers in the LFW database.

Unit Type	**Num. Hidden Layers**	**Pre-training**	**Num. Hidden Units**	**Acc. (%)**
Sigmoid	2 layer	no	1024	90.89
		yes	1024	91.50
	3 layer	no	1024	90.81
		yes	1024	91.12
ReLU	2 layer	no	512	91.22
		yes	512	91.83
	3 layer	no	512	91.37
		yes	512	92.08
ReLU+dropout	3 layer	no	512	92.13
		yes	512	**92.60**

perform better in all cases, as happened with the network with just one hidden layer. On the other hand, the use of pre-trained blocks usually improves the performance of the network, although this improvement is not very significant. Actually, this fact is in accordance with a current trend in the DL community that is forsaking the use of unsupervised pre-training to some specific tasks where the labelled data is scarce [LeCun et al., 2015]. Besides this issue, it is difficult to directly learn from the entire image using a fully-connected network in this task. The main reason of this problem is the large variability of the images due to changes in the lighting conditions, pose, expression, etc. Even adding more pre-trained layers the result is only marginally improved. Finally, note that the best result (92.60%) is obtained using a pre-trained network with 3 hidden layers trained including the dropout technique. This result serves as reference to compare the performance obtained with other networks.

5.3.3 Results with DCNN

In the previous section we have realized that using a deeper network with pre-trained layers only gives a modest improvement. This section evaluates a Deep Convolutional Neural Network (DCNN), which has shown great performance in other computer vision tasks [Hinton et al., 2012; Taigman et al., 2014].

The network architecture employed in the experiments is inspired by the excellent results obtained recently in [Taigman et al., 2014] with the LFW database in the face recognition problem. As explained in Section 2.2.4, DCNNs are composed alternating convolutional and pooling layers. More specifically, our network has a first convolutional layer (C1) with 32 filters of size 11×11, and its output is an image of size 143×143 (this is denoted by $32 \times 11 \times 11@143 \times 143$). The resulting 32 feature maps are fed into a max-pooling layer (P2) which takes the max over 3×3 spatial neighborhoods with a stride of 2. The second convolutional layer (C3) has 16 filters of size 9×9. After that, there is another max-pooling layer (P4) with the same parameters as before. After this part of the network, a fully-connected layer (F5) with 512 units plus a discriminative output layer are added at the end. A graphical representation of this network is depicted in Fig. 5.2.

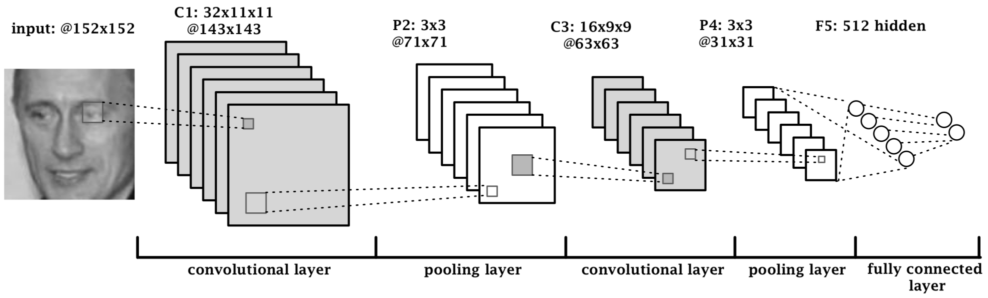


Figure 5.2. Graphical representation of a DCNN model. The parameters above denote the configuration used in the experiments of the gender recognition.

Before comparing the results obtained, it is also interesting to contrast the feature detectors obtained in the first layer of the network. The 32 learned filters are represented as images in Fig. 5.3. These filters are much simpler than those learned

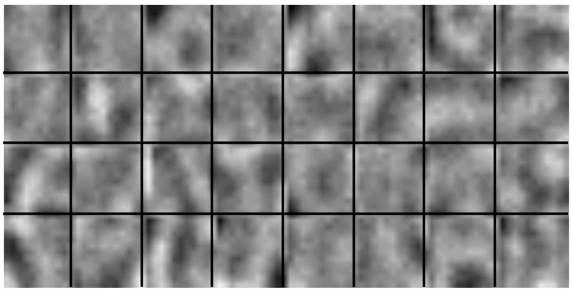


Figure 5.3. The 32 filters learned by the DCNN model in the first layer.

by the RBM (see Fig. 5.1), and are similar to other low-level handcrafted features such as Haar or Gabor wavelets. With regard to the type of units employed in the network, all the hidden units are ReLU due to the clear advantage compared with sigmoid units demonstrated before. Also, a 50% dropout has been applied only over the last fully-connected layer. This model has been evaluated for both databases, i.e. the LFW and the Gallagher. The results obtained are shown in Table 5.3. Note that the best result obtained with the DNN model for both datasets is also included. The parameters for the training process were selected using cross-validation in the LFW database, and they were also used with the Gallagher's database. As it can be seen,

Table 5.3. Accuracy on the test set for the DCNN model.

LFW Database	
Model	**Accuracy (%)**
Best DNN	92.60
DCNN	**94.09**

Gallagher's Database	
Model	**Accuracy (%)**
Best DNN	84.28
DCNN	**86.04**

there is a big improvement on the accuracy when using DCNN networks. This result confirms the good performance of mixing convolutional and pooling layers that are focused on learning from small regions of the image. This type of networks are able to take advantage of using deep architectures because the first layers are able to extract useful low-level features, while the upper ones learn to classify the samples according to their gender.

5.3.4 Results with Local-DNN

In this section we summarize the results obtained using our Local-DNN model presented in Chapter 4.

The Local-DNN model extracts several patches from the images to feed a neural network that learns to classify these patches. The extraction process is performed following the second method explained in Section 4.4.4. According to this method, a binary mask that aims to sweep the areas with high information content is created for each image (see Fig. 4.3). Using these masks, all the patches of size 13×13 pixels are extracted and normalized to have zero-mean and unit-variance. This size was inspired from other works that also use a local feature framework applied to face images, extracting patches similar to the size of an eye in the image ([Paredes et al., 2001]). At this point, it is important to stress the clear imbalance between the number of male and female images in the LFW database. For this reason we have randomly discarded some *male* patches from the training and from the validation set to have equally distributed folds during training. Obviously, the set of patches extracted from the test set remains unchanged. Furthermore, the network is composed of hidden layers with 512 ReLU units, and the number of layers has been changed to compare the performance.

All the results concerning these experiments are presented in three tables. The first one (Table 5.4) shows the accuracy at a patch level varying the number of hidden layers. With these results we can get an idea of how well the network is able to classify each patch as a male or as a female, i.e. to learn the complex mapping from the appearance of patches to classes. Note that the results displayed include those obtained including the location information in each local feature, as a modification of the base model (see Section 4.4.5).

According to these results, there is a big difference between the network with one hidden layer and the networks with two or more hidden layers. This issue occurs in both databases and denotes the poor learning capacity of the network with just one hidden layer. In this case, using deeper networks it does improve the performance in contrast to what happened feeding the entire image to the DNN (see Tables 5.1 and

Table 5.4. Accuracy at patch level on the test set for the Local-DNN model varying the number of hidden layers.

LFW Database			
Model	**Depth**	**Path Acc. (%)**	
		w/o loc.	**with loc.**
Local DNN	1 layer	68.48	71.18
	2 layer	74.30	77.17
	3 layer	74.50	**77.87**
	4 layer	74.34	77.26

Gallagher's Database			
Model	**Depth**	**Path Acc. (%)**	
		w/o loc.	**with loc.**
Local DNN	1 layer	64.87	67.14
	2 layer	70.70	**72.83**
	3 layer	70.51	72.65
	4 layer	70.52	72.37

5.2). On the other hand, it is also clear that including the location information of the patch in the local feature representation strengthens the learning of the network to better estimate the label of each patch.

Delving into the topic of the patch-based results, it is also interesting to analyze the distribution of this accuracy depending on the position of the patch in the image. To this end, Fig. 5.4 shows qualitatively the probability of accuracy depending on the position where the patch was extracted. In other words, a pixel in that image indicates how likely it is, that the patch extracted in that position predicts the correct class. In the image, the light color denotes higher probability and the dark color denotes lower probability. Note that the lowest probability is around 0.59 and the highest probability is around 0.78. These values are obtained with the best network in the LFW dataset. From this figure, it is clear that the *best* patches are those centered around the eyes and the mouth. To some extent, this makes sense because those areas of the face are very representative for the distinction between male and female.

Once the results at patch level have been presented, Table 5.5 resumes the accuracy obtained by applying the final decision rule to classify the whole face image. It is important to recall, that the Local-DNN model allows to make this decision using two methods according to the probabilistic framework explained in Section 4.4.2: the first one is summing all the posterior probabilities given by each patch of the image (see Eq. (4.12)), and the second one is based on a voting scheme where each patch chooses a class according to its maximum local posterior (see Eq. (4.13)). The results for both methods are presented in the table, varying the depth on the network as well. Note that the results obtained by summing posteriors use the same weight for all the contributions, so all the local features are equally reliable, i.e. $\alpha_i = 1/F$.

According to these results it is clear that summing posteriors yields slightly better results in all cases. Besides this fact, the results also show a big gap between

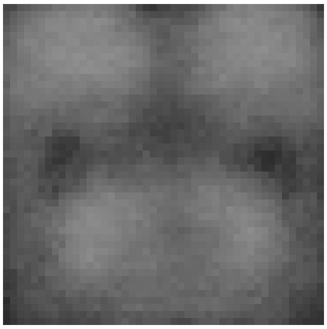


Figure 5.4. Probability of accuracy at patch level according to with the position where the patch was extracted. Light and dark colors denote high and low probability, respectively.

Table 5.5. Accuracy on the test set for our Local-DNN model varying the number of hidden layers.

LFW Database					
Model	**Depth**	**Acc. (%)**			
		$\sum$ Post.		**Voting**	
		w/o loc.	**with loc.**	**w/o loc.**	**with loc.**
Local DNN	1 layer	91.66	92.64	91.63	92.38
	2 layer	95.35	95.98	95.19	95.86
	3 layer	95.81	96.04	95.62	95.74
	4 layer	95.79	**96.25**	95.71	96.20

Gallagher's Database					
Model	**Depth**	**Acc. (%)**			
		$\sum$ Post.		**Voting**	
		w/o loc.	**with loc.**	**w/o loc.**	**with loc.**
Local DNN	1 layer	83.25	84.73	82.62	83.76
	2 layer	89.48	89.96	89.14	90.02
	3 layer	89.74	**90.58**	89.58	90.49
	4 layer	89.85	90.29	89.64	89.94

using a one hidden layer network or using more hidden layers. On the other hand, the improvement obtained at a patch level (see Table 5.4) by including the location information produces also an improvement at image level, obtaining the best results in both databases. It is also interesting to compare the accuracy obtained by the network at a patch-level and at image-level. The large difference between these two values (around 20%), denotes the strength of the method once all the contributions have been taken into account according to the probabilistic framework. This is the key of the method.

Besides these results, the modification of the base model regarding the use of different weights when summing posteriors during testing has been also evaluated (see Section 4.4.5). The α_i values are estimated using the probability of accuracy of each patch placement. These values were normalized to sum up 1.0 for all the patches extracted. This means that the higher α_i values correspond to local features with high probability of accuracy, and vice-versa. This modification was performed only for the best case in each database. The result obtained is 96.23% of accuracy in the LFW database and 90.23% in the Gallagher's database. Both results are quite similar to those obtained using equally reliable features in both databases. Probably, this is because the feature selection made by the binary mask of each image has already selected the most important zones from the image, and most of the patches selected by the binary mask are informative enough to be equally weighted.

Finally, Table5.6 presents cross-database results in order to show the validity of our approach and generalization capabilities. This results are obtained using one database for training and the other one for testing.

Table 5.6. Cross-database accuracy at image level using the Local-DNN model.

Train using LFW and test using Gallagher's					
Model	**Depth**	**Acc. (%)**			
		∑ Post.		**Voting**	
		w/o loc.	**with loc.**	**w/o loc.**	**with loc.**
Local DNN	1 layer	73.33	78.47	73.79	78.19
	2 layer	80.50	**83.03**	80.69	82.82
	3 layer	82.04	82.91	81.87	82.84
	4 layer	82.15	81.21	82.32	81.12

Train using Gallagher's and test using LFW					
Model	**Depth**	**Acc. (%)**			
		∑ Post.		**Voting**	
		w/o loc.	**with loc.**	**w/o loc.**	**with loc.**
Local DNN	1 layer	89.50	90.67	89.56	90.41
	2 layer	93.76	94.06	93.53	93.93
	3 layer	94.29	94.26	93.98	94.39
	4 layer	93.98	93.98	93.76	**94.48**

According to the results obtained, the Local-DNN model can generalize well to a different database at the expense of a small performance penalty. In addition, all the conclusions previously stated regarding the number of hidden layers, the use of the location information and the final decision rule method also apply in this case. It should be emphasized that our best cross-database results (83.03% and 94.48% of accuracy) are better than the only previously published cross-database results presented in [Dago-Casas et al., 2011], where a 81.02% and 89.77% were obtained when testing with the Gallagher's and LFW respectively.

5.3.5 Comparison of the results

In the previous section, all the results obtained with different networks have been summarized, including the Local-DNN model. In this section, these results are compared with other state-of-the-art results obtained with other methods on the same datasets. All of these results are included in Table 5.7.

Table 5.7. Best accuracy on the test set for DNN, DCNN and Local-DNN models, and other state-of-the-art results.

LFW Database	
Model	**Acc.(%)**
Best DNN	92.60
Best DCNN	94.09
Best Local-DNN	**96.25**
Gabor+PCA+SVM [Dago-Casas et al., 2011]	94.01
Boosted LBP+SVM [Shan, 2010]	94.44

Gallagher's Database	
Model	**Acc.(%)**
Best DNN	84.28
Best DCNN	86.04
Best Local-DNN	**90.58**
Gabor+PCA+SVM [Dago-Casas et al., 2011]	86.61
FPLBP+Drop SVM [Eidinger et al., 2014]	88.60
LBP+CH+SIFT+SVM [Fazl-Ersi et al., 2014]	**91.59**

According to these results, the Local-DNN model outperforms other DL methods, such as the DNN and the DCNN, and also obtains the best published result on the LFW dataset using all the images available. Note that other results not included in this table that use this database ([Ren and Li, 2014; Tapia and Perez, 2013]), were obtained removing many images, and using only 7443 and 6840 samples out of 13233, respectively. It is important to underline that the results presented in this section are obtained using the entire dataset, without removing any image. For this reason, both results cannot be compared on equal terms. On the other hand, [Fazl-Ersi et al., 2014] recently obtained a slightly better result in the Gallagher's database. However, this result is obtained using a complex ensemble composed of several handcrafted features such as LBPs, Color Histograms and SIFT, plus a SVM classifier. In contrast, the Local-DNN model is quite simple and generic to apply, and it also works well for other computer vision tasks as it has been shown in the experiments performed in Section 4.5.

5.4 Conclusions

This chapter presents several experiments performed using different DL networks on the gender recognition problem of face images. The target of the experiments is to evaluate the performance of these models using real-world images taken in unconstrained scenarios. To that end, two challenging face image datasets have been used: the Labelled Faces in the Wild (LFW) and the Groups/Gallagher dataset.

First of all, a standard DNN model has been evaluated varying the most important parameters of the model, such as the number of hidden layers, the type of hidden units and the use of pre-trained weights. The results with this type of model show that the use of deeper networks barely improves the performance, even with pre-training. Due to the large variability of the data, it is difficult to learn useful representations using fully-connected layer, i.e. when the learning is performed from the entire image. Regarding the type of units, it is clear that the ReLU units outperform the classical sigmoid function even with fewer number of neurons, which is a widespread conclusion among the DL researchers.

The next block of experiments is focused on models that take advantage of learning from small regions in the image. We have evaluated a DCNN that uses several layers of convolutional and pooling operations to learn useful feature detectors. The results obtained with this network substantially improves the result obtained with DNNs, and the network model actually benefits from using a deep architecture. This fact confirms the general perception demonstrated in the last years that most of the challenging computer vision problems can be better addressed using convolutional networks with ReLU units, and training the network with the dropout technique to prevent overfitting. Therefore, the use of unsupervised pre-training in discriminative problems has been relegated to problems where the labelled data is quite scarce [Bengio et al., 2013].

On the other hand, our novel Local-DNN model presented in Chapter 4 has also been evaluated. The results obtained with this model are even better than those obtained with the DCNN. Actually, the Local-DNN model obtains the best result published in the gender recognition problem with the LFW dataset using all the images, an it also obtains state-of-the-art results in the Gallagher dataset. The local-based learning used by this model works especially well in this specific task, where some kind of registration (scale) is applied to the images and some prior knowledge can be applied in order to select the most informative parts of the images. The performance can even be improved by including the location information where the patch was extracted in the local feature representation. It is also interesting to note the lack of improvement obtained when the local contributions are merged with different weights depending on the accuracy of each patch based on its location, as it already happened with the MNIST and the CIFAR-10 databases (see Section 4.5). This modification does not bring any improvement, so all the patches selected by the binary mask can be taken into account with the same weight.

Chapter 6

General Conclusions

This chapter aims to summarize the main conclusions extracted from the work presented in this thesis. The topic of this thesis is focused on the Deep Learning (DL) framework, which comprises an ensemble of models and methods to discover several representations of the data to be able to learn complex functions. More specifically, this thesis mainly presents two novel contributions related to this framework. On the one hand, Chapter 3 deals with different regularization techniques applied to the Restricted Boltzmann Machine model. Also, a new regularization scheme called Mask Selective Regularization (MSR) is presented and evaluated. On the other hand, Chapter 4 presents a discriminative model called Local Deep Neural Network (Local-DNN), which takes advantage of learning from small regions in images by using deep architectures. The specific conclusions on these methods are detailed in Sections 6.1 and 6.2 of this chapter.

These contributions have been made possible thanks to the previous extensive work carried out to have the necessary knowledge in this area within the Machine Learning context. Some of this knowledge is summarized in Chapter 2 as an overview of the most important models and methods within the DL framework. These methods have dramatically improved the state-of-the-art in several tasks and domains for the last years. Inspired by this success and given our special interest in biometric applications, Chapter 5 summarizes the results obtained using several DL methods in the specific task of the gender recognition using face images. According to the results obtained, our Local-DNN model performed remarkably well in two face datasets evaluated, improving other state-of-the-art techniques in some cases.

Finally, it is important to highlight that part of the work presented in this thesis has provided a basis for opening a new line of research in our group at the university. Let us hope that this work will help other researchers to continue working on these topics.

6.1 Conclusions on Regularization Methods for RBMs

Chapter 3 studies several regularization methods for the RBM model. According to the literature there are two typical regularization techniques commonly applied to the parameters (weights) of the RBM during training: one based on the L_2 norm, and the other one based on the L_1 norm. Both regularizations have their own advantages and disadvantages. Therefore, we propose a novel regularization method, called Mask Selective Regularization (MSR), that aims at applying both norms in an effective way. Each connection of the RBM is regularized with just one of these norms, so that the L_1 norm is applied to the weak connections to force real zeros, and the L_2 is applied to the meaningful connections to avoid them to grow too much. The selection of the connections is performed adaptively by using a binary mask that takes into account the topology of the input data and the current state of the weights during the learning process.

In the experiments performed with several datasets, the discriminative performance of a neural network pre-trained with RBMs is compared changing the regularization method applied during the pre-training. The main conclusions drawn from these experiments can be summarized in the following:

- The networks trained with the MSR technique obtain better results in most of the cases evaluated, specially when the input data has been corrupted with different types of noise to simulate possible sources of error.
- The MSR method can be applied successfully with non-image data, such as text, and it can be used in upper layers of the deep network as well. This is in contrast to other methods published in the literature that assume local statistics of the data (like in images) to decide which connections should be removed, and they cannot be applied in the upper weights of the network.
- The qualitative results given by the histogram of the weights and the feature detectors obtained reveals that the MSR gets what it was initially proposed, i.e. to force real zeros in the weak connections and to avoid the remaining ones to grow too much.

6.2 Conclusions on the Local-DNN model

Chapter 4 presents a novel discriminative model called Local Deep Neural Network (Local-DNN). This model is based on two key concepts: local features and deep architectures. Several local windows (patches) are extracted at different locations of the image, and a DNN learns to classify these patches according to the label of the image to which it belongs. In the testing step, all these local contributions are merged to predict a single label for the image.

Several experiments using different databases have been performed, varying also the most important parameters of the model. Also, the results are compared with those obtained using a standard DNN to show the advantages of restricting the learning to local regions in the image. The main conclusions drawn from these experiments are summarized below:

- The experiments performed using challenging datasets, such as the CIFAR-10 or the datasets employed in the gender recognition task (LFW and Gallagher), show the lower performance obtained with a standard DNN that learns from the entire image. Moreover, in these cases there is no significant benefit of using deeper networks.
- In contrast, the Local-DNN model works remarkably well in the cases cited above, specially in the gender recognition problem where some registration has been applied to the images and where some prior knowledge can be applied in order to select the most informative parts. Note that Local-DNN obtains the best result published in the LFW dataset and the second best result in the Gallagher dataset.
- In a simpler task like MNIST, the Local-DNN does not take advantage of learning from small patches. However, extraordinary results are obtained with this model by learning from larger patches and merging these contributions during testing. These patches are no longer *local*, and actually they can be seen as several translated versions of the original images.
- The performance of the model can be improved when the location information where the patch was extracted is included in the local feature representation.
- The fusion method based on summing posteriors works slightly better than the method based on a direct voting scheme.
- Using a weighted sum of contributions does not improve the results when these weights are computed according to the accuracy of each patch depending on its location. Therefore, similar results are obtained if all the patches available are taken into account with the same weight.
- The main advantages of the model presented are given by its simplicity and generalization capabilities. It should be also highlighted that the model might work well in discriminative tasks where there are few samples per class, in contrast to DCNNs that usually tend to overfit the training data, even with regularization techniques such as dropout.

6.3 Directions for Future Research

This section briefly offers some pending issues and possible future directions for research regarding the DL methods presented in this thesis. The MSR technique has shown to work well as a regularization during the pre-training. However, the binary masks that split both regularizations are not taken into account during the fine-tuning process. It would make sense to keep also these binary masks when the Backpropagation algorithm is used to modify all the weights of the discriminative network together. This idea is somehow similar to that proposed in [Wan et al., 2013], where several random connections are removed during training according to the different binary masks obtained. Besides this line of research, some effort should be made to better evaluate the MSR method in other tasks involving non image data, such as the text classification.

The Local-DNN model has made possible to assess the importance of learning from small regions in challenging tasks dealing with images. The results obtained are quite encouraging given the simplicity of the model. For this reason, it would be interesting to use a more powerful model like a convolutional network to further exploit the local-based learning proposed. The performance of this model should be also addressed to other tasks where the data available is scarce. In this scenario, the Local-DNN model might perform well because the network would not overfit the training data thanks to the increase of the data samples given by the patches extracted. Finally, it would be interesting to combine several patches of different sizes in the fusion step, especially in other tasks where the objects in the images are displayed at very different scales. In this case, several networks must be trained using different patch sizes, and each of them would specialize in one specific scale.

6.4 Dissemination

Dissemination of findings related to the investigation in this thesis has extended to various publications in scientific journals and international conferences. In this section, we enumerate these publications pointing out their relation with the thesis.

In the beginning of the PhD, most of the work was focused on several Machine Learning areas, especially those related to the face recognition and the point of regard estimation problems. This work served as the main topic for my Master Thesis ([Mansanet, 2012]), and the results obtained were published in the following international conference:

- **Mansanet, J.**, Albiol, A., Paredes, R., Mossi, J.M., Albiol, A. (2013). Estimating Point of Regard with a consumer camera at a distance. *6th Iberian Conference, IbPRIA 2013, Funchal, Madeira, Portugal, June 5-7, 2013. Proceedings*, pages 881–888.

Regarding to the RBM model, much work has been done . On the one hand, several experiments using this model as a feature extractor in the gender recognition problem were performed to acquire all the knowledge and background exposed in Section 2.3.1. The results obtained were published in the following international conference:

- **Mansanet, J.**, Albiol, A., Paredes, R., Villegas, M., Albiol, A. (2014). Restricted Boltzmann Machines for Gender Recognition. *11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part I*, pages 274–281.

On the other hand, part of the work exposed in Chapter 3 about the novel regularization method MSR has been published in a very prestigious international journal:

- **Mansanet, J.**, Albiol, A., Paredes, R., Albiol, A. (2015). Mask selective regularization for restricted Boltzmann machines. *Neurocomputing, Volume 165, 1 October 2015*, pages 375–383.

Finally, the description of the Local-DNN model (see Chapter 4) and several experiments summarized in Chapter 5, are included in one article in the prestigious *Pattern Recognition Letters* journal. This article is currently accepted and it will be published in the near future.

- **Mansanet, J.**, Albiol, A., Paredes, R. (2015). Local Deep Neural Networks for gender recognition. *Pattern Recognition Letters, DOI: 10.1016/j.patrec.2015.11.015.*

Besides the research articles published, the work carried out during this thesis has allowed to actively contribute to the organization of the *First AERFAI Autumn School on Deep Learning (AASDL)*. This event covered both theoretical and practical issues of Deep Learning methods with several lectures. More details about this school can be found at [AERFAI, 2015].

Appendix A

Public Databases and Evaluation Protocols

This appendix describes the databases and their related evaluation protocol used for the experiments performed in this thesis. These datasets are publicly available for researchers, and conform a great variety of data types.

A.1 MNIST Database

The MNIST database[1] is composed of images of handwritten digits. It is a subset of a larger set available from NIST. The original black and white images were size-normalized to fit in a 20×20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28×28 pixels image by computing the center of mass of the pixels. This database is widely evaluated using Machine Learning and Pattern Recognition methods because provides a real-world data while spending minimal efforts on preprocessing and formatting. Some examples of the database are displayed in Fig. A.1.

The evaluation method fixes a training set of 60,000 digits and a test set of 10,000 digits. For validation purposes it is typical to use 50,000, equally distributed random images for training and the remaining 10,000 for validation.

A.2 USPS Database

The US Postal Service (USPS) database is composed of images of handwritten digits. It contains normalized gray scale images of size 16×16 pixels. Some examples of the database are displayed in Fig. A.2. In contrast to the MNIST dataset, these digits are normalized to fit the available area of the entire image.

[1]MNIST database is available here: http://yann.lecun.com/exdb/mnist/.

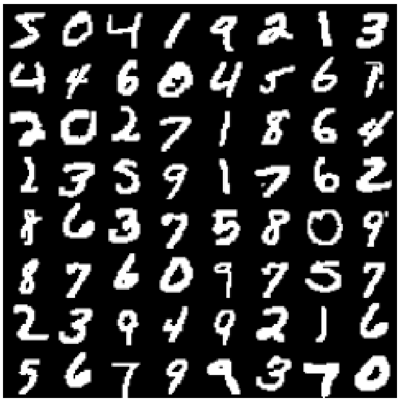


Figure A.1. Some examples of handwritten digits of the MNIST dataset.

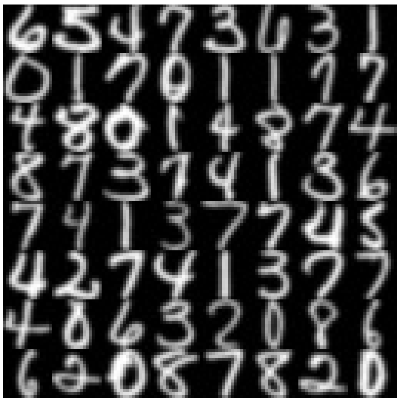


Figure A.2. Some examples of handwritten digits of the USPS dataset.

There is not an official evaluation method for this dataset, but a known splitting of 7291 cases for training and 2007 for testing has been traditionally used[2]. The experiments performed in this thesis have followed this partition.

A.3 20 Newsgroup Database

The 20 Newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups. This dataset has become very popular for experiments regarding text classification and text clustering tasks. Table 2 shows a list of the 20 newsgroups partitioned (more or less) according to subject matter.

Table A.1. The 20 classes of the 20 Newsgroup dataset, partitioned according to the subject matter.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x	rec.autos rec.motorcycles rec.sport.baseball rec.sport.hockey	sci.crypt sci.electronics sci.med sci.space
misc.forsale	talk.politics.misc talk.politics.guns talk.politics.mideast	talk.religion.misc alt.atheism soc.religion.christian

For the experiments performed in this thesis we have used a version of this dataset[3] which contains 11269 training samples and 7505 test samples after some preprocessing steps. The samples of this version were preprocessed as follows. First, the original data was tokenized using the *Rainbow* toolkit [McCallum, 1996]. After that, the data was converted to Matlab format using the word counts from a fixed vocabulary with the 5,000 most informative words. Therefore, each sample is represented by a feature vector with 5,000 values.

A.4 CIFAR-10 Database

The CIFAR-10 dataset[4] is a subset of the Tiny Images dataset which contains 60,000 images divided among 10 classes. The size of the images is 32×32 pixels, and all of them were collected from the web representing natural scenes. This dataset is highly varied because there is not a canonical viewpoint or scale at which the objects appear. The only criteria for including an image was that it must contain one dominant instance of the available classes, so that the object in the image must be

[2]USPS dataset is available here: http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html.
[3]20Newsgroup dataset is available here: http://qwone.com/∼jason/20Newsgroups/20news-bydate-matlab.tgz.
[4]CIFAR-10 dataset is available here: http://www.cs.toronto.edu/∼kriz/cifar.html.

easily identifiable as belonging to the class indicated by the image label. The classes are completely mutually exclusive. Some examples of the database are displayed in Fig. A.3. Each row represents one of the 10 classes: *airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck.*

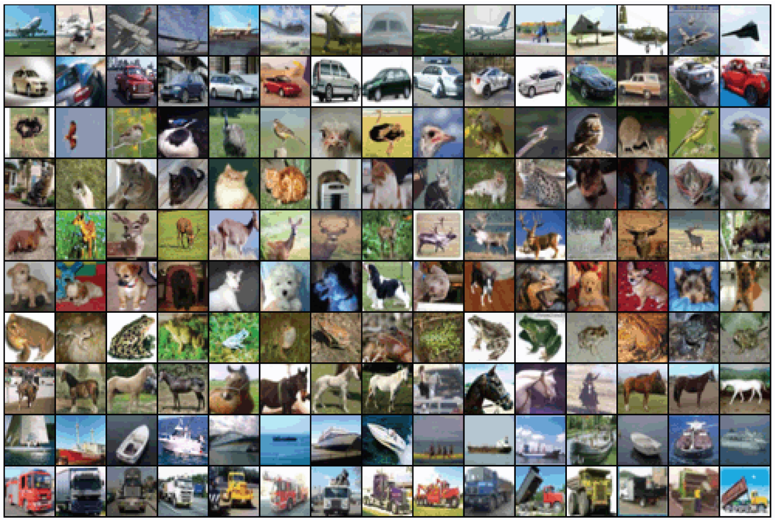


Figure A.3. Some examples of the CIFAR-10 dataset images.

The evaluation protocol proposed for this dataset splits the data into 50,000 training images and 10,000 test images. Both sets contain the same number of images in each class.

A.5 Labelled Faces in the Wild Database

The Labelled Faces in the Wild database (LFW) [Huang et al., 2007] is composed of face images, and it is meant to study the problem of face recognition in unconstrained conditions. The face images were collected from the web, and the only constraint on these faces is that they were detected by the Viola-Jones face detector. The database contains 13,233 color images (10,256 male and 2,977 female) from 5,749 celebrities. Some examples of images of the database are displayed in Fig. A.4. The size of these images is 250×250 pixels. Due to the unconstrained nature of the original images, in this thesis we have worked with an aligned version of the database[5], which is composed of the same images, but the faces are aligned to a canonical pose.

[5]Aligned LFW dataset with deep funneling is available here: http://vis-www.cs.umass.edu/lfw/lfw-deepfunneled.tgz.

Figure A.4. Some examples of the LFW dataset images.

In this thesis we have used this database for performing experiments on the gender recognition problem. To that end we have followed a benchmark proposed by the Facial Image Processing and Analysis group (FIPA, `http://fipa.cs.kit.edu`), which splits the original data in 5 folds[6]. The evaluation protocol suggest to use 4 folds for training and 1 for testing. This should be done for all the combinations, and the results are averaged.

A.6 Groups/Gallagher Database

The Groups Database [Gallagher and Chen, 2009] is composed of 5,080 images containing 28,231 faces labeled with the age and gender information. This database is best known as the *Gallagher's database* because of its main contributor Andrew C. Gallagher. All the images were collected from Flickr in very different acquisition conditions, so the images are really challenging, containing significant pose, illumination, expression, age and ethnic differences. Some examples of the original images of the database representing groups of people are displayed in Fig. A.5.

Figure A.5. Some examples of the Gallagher dataset images.

[6]The 5 folds proposed by FIPA are available here http://fipa.cs.kit. edu/download/LFW- gender-folds.dat.

In this thesis we have used this dataset for performing experiments on the gender recognition problem. There is not an standard benchmark for gender classification using this database, so we have followed the protocol proposed by [Dago-Casas et al., 2011]. According to this, a new version of the dataset is created by removing those images whose interocular distance was less than 20 pixels (to eliminate low resolution images). Some of the male faces were also discarded to have an equal number of male and female faces. This results in a final dataset containing 5 equally sized folds containing altogether 14760 images[7]. For the evaluation process proposed, 4 folds are used for training and the remaining one for testing. At the end, all the combinations are averaged.

[7]The 5 folds with the images selected are available here http://fipa.cs.kit.edu/download/Gallagher_gender_5folds.txt.

Bibliography

AERFAI, PRHLT, U. (2015). https://www.upv.es/contenidos/AASDL/. (Cited on page 93)

Baluja, S. and Rowley, H. A. (2007). Boosting sex identification performance. *Int. J. Comput. Vision*, 71(1):111–119. (Cited on page 75)

Bartle, A. and Zheng, J. (2015). Gender classification with deep learning. Technical report, The Stanford NLP Group. (Cited on page 76)

Bellman, R. (1957). *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA. (Cited on page 2)

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. (Cited on pages 3, 16, and 23)

Bengio, Y. and Bengio, S. (2000). Modeling high-dimensional discrete data with multi-layer neural networks. In *Advances in Neural Information Processing Systems 12*, pages 400–406. (Cited on page 8)

Bengio, Y. and Courville, A. (2013). Deep learning of representations. In *Handbook on Neural Information Processing*, pages 1–28. Springer. (Cited on page 2)

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8). (Cited on pages 2 and 86)

Bengio, Y. and Glorot, X. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS 2010*, volume 9, pages 249–256. (Cited on pages 10 and 77)

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2007). Greedy layer-wise training of deep networks. In Schölkopf, B., Platt, J., and Hoffman, T., editors, *Advances in Neural Information Processing Systems 19*, pages 153–160. MIT Press. (Cited on page 22)

Bo, L., Ren, X., and Fox, D. (2010). Kernel Descriptors for Visual Recognition. In *Advances in Neural Information Processing Systems*. (Cited on page 66)

Buchala, S., Davey, N., Frank, R., and Gale, T. (2004). Dimensionality reduction of face images for gender classification. *Intelligent Systems, 2004. Proceedings. 2004 2nd International IEEE Conference*, 1:88–93 Vol.1. (Cited on page 75)

Cai, X., Hu, S., and Lin, X. (2012). Feature extraction using Restricted Boltzmann Machine for stock price prediction. In *Computer Science and Automation Engineering (CSAE), 2012 IEEE International Conference on*, pages 80–83. (Cited on page 28)

Carreira-Perpinan, M. A. and Hinton, G. E. (2005). On contrastive divergence learning. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 33–40. (Cited on page 19)

Chan, T., Jia, K., Gao, S., Lu, J., Zeng, Z., and Ma, Y. (2014). Pcanet: A simple deep learning baseline for image classification? *CoRR*. (Cited on pages 66, 70, and 71)

Chandra, S., Kumar, S., and Jawahar, C. V. (2013). Learning multiple non-linear sub-spaces using k-rbms. In *Computer Vision and Pattern Recognition*. (Cited on page 54)

Cho, K., Ilin, A., and Raiko, T. (2012). Tikhonov-type regularization for restricted boltzmann machines. In *Artificial Neural Networks and Machine Learning - ICANN 2012*, volume 7552, pages 81–88. Springer. (Cited on page 29)

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22(12):3207–3220. (Cited on pages 70 and 71)

Coates, A., Karpathy, A., and Ng, A. Y. (2012). Emergence of object-selective features in unsupervised feature learning. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2681–2689. Curran Associates, Inc. (Cited on page 54)

Coates, A., Lee, H., and Ng, A. Y. (2011). An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*. (Cited on pages 47, 53, and 66)

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems (MCSS)*, 2(4):303–314. (Cited on page 10)

Dago-Casas, P., González-Jiménez, D., Yu, L. L., and Alba-Castro, J. L. (2011). Single- and cross- database benchmarks for gender classification under unconstrained settings. In *ICCV Workshops*, pages 2152–2159. IEEE. (Cited on pages 74, 75, 76, 85, and 100)

Dahl, G., aurelio Ranzato, M., rahman Mohamed, A., and Hinton, G. E. (2010). Phone recognition with the mean-covariance restricted boltzmann machine. In Lafferty, J., Williams, C., Shawe-Taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 469–477. Curran Associates, Inc. (Cited on page 23)

Dayan, P., Hinton, G. E., Neal, R. N., and Zemel, R. S. (1995). The Helmholtz machine. *Neural Computation*, 7:889–904. (Cited on page 21)

Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Mach. Learn.*, 46(1-3):161–190. (Cited on pages 70 and 71)

Deriche, R. and Giraudon, G. (1993). A computational approach for corner and vertex detection. *Int. J. Comput. Vision*, 10(2):101–124. (Cited on page 59)

Eidinger, E., Enbar, R., and Hassner, T. (2014). Age and gender estimation of unfiltered faces. *IEEE Transactions on Information Forensics and Security*, 9(12):2170 – 2179. (Cited on pages 75 and 85)

Erhan, D., Courville, A., Bengio, Y., and Vincent, P. (2010). Why does unsupervised pre-training help deep learning? In *Proceedings of AISTATS 2010*, volume 9, pages 201–208. (Cited on page 23)

Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2013). Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929. (Cited on page 56)

Fazl-Ersi, E., Mousa-Pasandi, M., Laganiere, R., and Awad, M. (2014). Age and gender recognition using informative features of various types. In *IEEE ICIP 2014*. (Cited on pages 85 and 86)

Fischer, A. and Igel, C. (2012). An introduction to restricted boltzmann machines. In *CIARP*, volume 7441 of *Lecture Notes in Computer Science*, pages 14–36. Springer. (Cited on page 26)

Freund, Y. and Haussler, D. (1991). Unsupervised learning of distributions of binary vectors using 2-layer networks. In *Advances in Neural Information Processing Systems 4*, pages 912–919. (Cited on page 16)

Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36:193–202. (Cited on page 14)

Gallagher, A. and Chen, T. (2009). Understanding images of groups of people. In *Proc. CVPR.* (Cited on page 99)

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741. (Cited on page 18)

Glorot, X., Bordes, A., and Bengio, Y. (2010). Deep sparse rectifier networks. NIPS*2010 Workshop on Deep Learning and Unsupervised Feature Learning (poster). (Cited on pages 11, 23, 48, 61, 66, and 77)

Golik, P., Doetsch, P., and Ney, H. (2013). Cross-entropy vs. squared error training: a theoretical and experimental comparison. In *Interspeech*, pages 1756–1760, Lyon, France. (Cited on page 12)

Goodfellow, I. J., Warde-Farley, D., Mirza, M., Courville, A., and Bengio, Y. (2013). Maxout networks. In *ICML*. (Cited on pages 11, 48, 66, and 70)

Graf, A. B. A. and Wichmann, F. A. (2002). Gender classification of human faces. In *Proceedings of the 2nd International Workshop on Biologically Motivated Computer Vision (BMCV)*, pages 491–500, London, UK. (Cited on page 75)

Gregor, K. and LeCun, Y. (2010). Emergence of complex-like cells in a temporal product network with local receptive fields. *CoRR*, abs/1006.0448. (Cited on page 54)

Gülçehre, Ã. and Bengio, Y. (2013). Knowledge matters: Importance of prior information for optimization. *CoRR*. (Cited on page 54)

han Lin, T. and Kung, H. T. (2014). Stable and efficient representation learning with nonnegativity constraints. In Jebara, T. and Xing, E. P., editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1323–1331. JMLR Workshop and Conference Proceedings. (Cited on page 52)

Hinton, G., Dayan, P., Dayan, B. J., and Neal, R. M. (1995). The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268:1158–1161. (Cited on pages 21 and 22)

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Comput.*, 14(8):1771–1800. (Cited on pages 16, 18, and 19)

Hinton, G. E. (2007). Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434. (Cited on page 23)

Hinton, G. E. (2010). A practical guide to training restricted boltzmann machines. Technical report, University of Toronto. (Cited on pages 20 and 26)

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554. (Cited on pages 8, 22, 23, 27, 28, and 44)

Hinton, G. E. and Salakhutdinov, R. (2009). Replicated softmax: an undirected topic model. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1607–1614. Curran Associates, Inc. (Cited on page 28)

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313:504–507. (Cited on pages 8, 16, 20, 22, 28, 40, 41, 67, 70, and 71)

Hinton, G. E. and Salakhutdinov, R. R. (2012). A better way to pretrain deep boltzmann machines. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2447–2455. Curran Associates, Inc. (Cited on page 23)

Hinton, G. E. and Sejnowski, T. J. (1986). Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*, chapter Learning and Relearning in Boltzmann Machines, pages 282–317. MIT Press. (Cited on page 16)

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580. (Cited on pages 66 and 79)

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. (Cited on pages 10 and 12)

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366. (Cited on page 10)

Huang, G. B., Lee, H., and Learned-Miller, E. G. (2012a). Learning hierarchical representations for face verification with convolutional deep belief networks. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 2518–2525. (Cited on pages 23 and 74)

Huang, G. B., Mattar, M., Lee, H., and Learned-Miller, E. (2012b). Learning to align from scratch. In *NIPS*. (Cited on page 76)

Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst. (Cited on page 98)

Huang, J. and Mumford, D. (1999). Statistics of natural images and models. In *CVPR*, pages 1541–1547. IEEE Computer Society. (Cited on page 29)

Ji, N., Zhang, J., Zhang, C., and Yin, Q. (2014). Enhancing performance of restricted boltzmann machines via log-sum regularization. *Knowledge-Based Systems*, 63:82–96. (Cited on page 28)

Jiang, Y., Li, S., Li, P., and Dai, Q. (2014a). Multi-feature deep learning for face gender recognition. In *Information Technology and Artificial Intelligence Conference (ITAIC)*, pages 507–511. (Cited on page 76)

Jiang, Z., Guo, P., and Peng, L. (2014b). Locality-constrained low-rank coding for image classification. (Cited on page 46)

Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. Master's thesis, Department of Computer Science, University of Toronto. (Cited on pages 14, 17, 20, 48, 52, 54, and 66)

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc. (Cited on pages 8 and 15)

Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. (2009). Attribute and Simile Classifiers for Face Verification. In *IEEE International Conference on Computer Vision (ICCV).* (Cited on page 75)

Larochelle, H. and Bengio, Y. (2008). Classification using discriminative restricted boltzmann machines. In *In ICML '08: Proceedings of the 25th international conference on Machine learning. ACM.* (Cited on pages 16 and 47)

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444. (Cited on pages 7 and 79)

LeCun, Y., Boser, B., Denker, J. S., Howard, R. E., Habbard, W., Jackel, L. D., and Henderson, D. (1990). Handwritten digit recognition with a back-propagation network. In Touretzky, D. S., editor, *Advances in neural information processing systems 2*, pages 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. (Cited on page 44)

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324. (Cited on pages 8, 14, and 28)

Lee, C., Xie, S., Gallagher, P., Zhang, Z., and Tu, Z. (2015). Deeply-supervised nets. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2015, San Diego, California, USA, May 9-12, 2015.* (Cited on pages 66 and 70)

Lee, H., Ekanadham, C., and Ng, A. Y. (2008). Sparse deep belief net model for visual area v2. In *Advances in Neural Information Processing Systems 20*. MIT Press. (Cited on pages 27 and 28)

Lee, H., Grosse, R., Ranganath, R., and Ng, A. Y. (2009a). Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th International Conference on Machine Learning*, pages 609–616. (Cited on page 23)

Lee, H., Largman, Y., Pham, P., and Ng, A. Y. (2009b). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in Neural Information Processing Systems 22*, pages 1096–1104. Curran Associates, Inc. (Cited on page 23)

Lee, H., Pham, P., Largman, Y., and Ng, A. Y. (2009c). Unsupervised feature learning for audio classification using convolutional deep belief networks. In Bengio, Y., Schuurmans, D., Lafferty, J., Williams, C., and Culotta, A., editors, *Advances in Neural Information Processing Systems 22*, pages 1096–1104. Curran Associates, Inc. (Cited on page 28)

Lee, T. S., Mumford, D., Romero, R., and Lamme, V. A. (1998). The role of the primary visual cortex in higher level vision. *Vision Research*, 38(15 \16):2429–2454. (Cited on page 3)

Leng, X. and Wang, Y. (2008). Improving generalization for gender classification. In *ICIP*, pages 1656–1659. (Cited on page 75)

Levi, G. and Hassner, T. (2015). Age and gender classification using convolutional neural networks. In *IEEE Conf. on CVPR workshops*. (Cited on page 76)

Li, J., Zhao, B., and Zhang, H. (2009). Face recognition based on pca and lda combination feature extraction. In *Proceedings of the 2009 First IEEE International Conference on Information Science and Engineering*, ICISE '09, pages 1240–1243, Washington, DC, USA. IEEE Computer Society. (Cited on page 51)

Li, W. (1990). Mutual information functions versus correlation functions. *Journal of Statistical Physics*, 60:823–837. (Cited on page 32)

Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *CoRR*, abs/1312.4400. (Cited on pages 54, 66, and 70)

Luo, H., Shen, R., Niu, C., and Ullrich, C. (2011). Sparse group restricted boltzmann machines. In Burgard, W. and Roth, D., editors, *AAAI*. AAAI Press. (Cited on page 28)

Mansanet, J. (2012). Técnicas de regresión para la estimación de la localización de la mirada. Master's thesis, Universitat Politéctiva de Valéncia. (Cited on page 92)

Mansanet, J., Albiol, A., and Paredes, R. (2015a). Local deep neural networks for gender recognition. *Pattern Recognition Letters*. (Not cited)

Mansanet, J., Albiol, A., Paredes, R., and Albiol, A. (2015b). Mask selective regularization for restricted boltzmann machines. *Neurocomputing*, 165:375–383. (Not cited)

Mansanet, J., Albiol, A., Paredes, R., Mossi, J. M., and Albiol, A. (2013). Estimating point of regard with a consumer camera at a distance. In *IbPRIA*, pages 881–888. (Not cited)

Mansanet, J., Albiol, A., Paredes, R., Villegas, M., and Albiol, A. (2014). Restricted boltzmann machines for gender classification. In *Image Analysis and Recognition - 11th International Conference, ICIAR 2014, Vilamoura, Portugal, October 22-24, 2014, Proceedings, Part I*, pages 274–281. (Not cited)

McCallum, A. K. (1996). Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow. (Cited on page 97)

Moghaddam, B. and Yang, M.-H. (2002). Learning gender with support faces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):707–711. (Cited on page 75)

Mohr, R., Picard, S., and Schmid, C. (1997). Bayesian decision versus voting for image retrieval. In Sommer, G., Daniilidis, K., and Pauli, J., editors, *CAIP*, volume 1296 of *Lecture Notes in Computer Science*, pages 376–383. Springer. (Cited on page 51)

Müller, A., Schulz, H., and Behnke, S. (2010). Topological features in locally connected rbms. In *IJCNN*, pages 1–6. IEEE. (Cited on page 33)

Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress. (Cited on pages 20, 21, 28, 53, and 77)

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04. ACM. (Cited on pages 27 and 30)

Ng, C., Tay, Y., and Goi, B.-M. (2012a). Recognizing human gender in computer vision: A survey. In Anthony, P., Ishizuka, M., and Lukose, D., editors, *PRICAI 2012: Trends in Artificial Intelligence*, volume 7458 of *Lecture Notes in Computer Science*, pages 335–346. Springer Berlin Heidelberg. (Cited on page 75)

Ng, C. B., Tay, Y. H., and Goi, B.-M. (2012b). Vision-based human gender recognition: A survey. *CoRR*, abs/1204.1611. (Cited on page 73)

Ngiam, J., Chen, Z., Chia, D., Koh, P. W., Le, Q. V., and Ng, A. Y. (2010). Tiled convolutional neural networks. In Lafferty, J., Williams, C., Shawe-taylor, J., Zemel, R., and Culotta, A., editors, *Advances in Neural Information Processing Systems 23*, pages 1279–1287. Curran Associates, Inc. (Cited on page 54)

Nielsen, M. (2015). Neural Networks and Deep Learning. http://neuralnetworksanddeeplearning.com. (Cited on pages 10 and 12)

Paredes, R., Pérez, J. C., Juan, A., and Vidal, E. (2001). Local representations and a direct voting scheme for face recognition. In *In Workshop on Pattern Recognition in Information Systems*, pages 71–79. (Cited on pages 51, 54, 60, and 81)

rahman Mohamed, A., Dahl, G. E., and Hinton, G. E. (2009). Deep belief networks for phone recognition. In *NIPS Workshop on Deep Learning for Speech Recognition and Related Applications*. (Cited on page 28)

Ren, H. and Li, Z. (2014). Gender recognition using complexity-aware local features. In *22nd ICPR 2014*, pages 2389–2394. (Cited on page 86)

Rifai, S., Bengio, Y., Courville, A. C., Vincent, P., and Mirza, M. (2012). Disentangling factors of variation for facial expression recognition. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part VI*, pages 808–822. (Cited on page 23)

Rowley, H., Baluja, S., and Kanade, T. (1996). Neural network-based face detection. In *Computer Vision and Pattern Recognition '96*. (Cited on page 54)

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning internal representations by error propagation. In Anderson, J. A. and Rosenfeld, E., editors, *Neurocomputing: Foundations of Research*, pages 673–695. MIT Press, Cambridge, MA, USA. (Cited on pages 7, 8, and 11)

Ruprah, T. (2012). *Face Recognition Using Pca and Lda Algorithm.* Lap Lambert Academic Publishing GmbH KG. (Cited on page 51)

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, pages 1–42. (Cited on page 15)

Salakhutdinov, R. and Hinton, G. (2009a). Deep boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455. (Cited on pages 23 and 41)

Salakhutdinov, R. and Hinton, G. (2009b). Deep Boltzmann machines. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, volume 5, pages 448–455. (Cited on pages 70 and 71)

Salakhutdinov, R., Mnih, A., and Hinton, G. (2007). Restricted boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, pages 791–798. ACM. (Cited on page 28)

Sarikaya, R., Hinton, G., and Deoras, A. (2014). Application of deep belief networks for natural language understanding. *IEEE Transactions on Audio Speech and Language Processing.* (Cited on page 23)

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning*, 5(2):197–227. (Cited on page 58)

Schmid, C. and Mohr, R. (1997). Local grayvalue invariants for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(5):530–535. (Cited on page 51)

Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 3642–3649. (Cited on pages 66 and 70)

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832. (Cited on page 74)

Schulz, H., Müller, A., and Behnke, S. (2011). Exploiting local structure in boltzmann machines. *Neurocomputing*, 74(9):1411–1417. (Cited on page 29)

Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., and LeCun, Y. (2014). Overfeat: Integrated recognition, localization and detection using convolutional networks. In *International Conference on Learning Representations (ICLR 2014)*. CBLS. (Cited on page 8)

Sermanet, P., Kavukcuoglu, K., Chintala, S., and LeCun, Y. (2013). Pedestrian detection with unsupervised multi-stage feature learning. In *Proc. International Conference on Computer Vision and Pattern Recognition (CVPR'13)*. IEEE. (Cited on page 23)

Shakhnarovich, G., Viola, P. A., and Moghaddam, B. (2002). A unified learning framework for real time face detection and classification. In *Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition*, pages 16–. IEEE Computer Society. (Cited on page 75)

Shan, C. (2010). Gender classification on real-life faces. In *ACIVS (2)*, pages 323–331. (Cited on pages 75 and 85)

Shan, C. (2012). Learning local binary patterns for gender classification on real-world face images. *Pattern Recognition Letters*, 33(4):431 – 437. (Cited on pages 74 and 75)

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In Rumelhart, D. E., McClelland, J. L., and PDP Research Group, C., editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1*, volume 1, pages 194–281. MIT Press. (Cited on page 16)

Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In Pereira, F., Burges, C., Bottou, L., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 25*, pages 2951–2959. Curran Associates, Inc. (Cited on page 52)

Sohn, K. and Lee, H. (2012). Learning invariant representations with local transformations. In Langford, J. and Pineau, J., editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1311–1318, New York, NY, USA. ACM. (Cited on page 52)

Song, L., Langfelder, P., and Horvath, S. (2012). Comparison of co-expression measures: mutual information, correlation, and model based indices. *BMC Bioinformatics*, 13:328. (Cited on page 32)

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. (Cited on pages 15, 53, 54, and 79)

Srivastava, N. and Salakhutdinov, R. (2014). Multimodal learning with deep boltzmann machines. *Journal of Machine Learning Research*, 15:2949–2980. (Cited on page 23)

Susskind, J., Anderson, A., Hinton, G., and Movellan, J. (2008). *Generating Facial Expressions with Deep Belief Nets*. INTECH Open Access Publisher. (Cited on page 23)

Sutskever, I. and Tieleman, T. (2010). On the convergence properties of contrastive divergence. In Teh, Y. W. and Titterington, D. M., editors, *AISTATS*, volume 9 of *JMLR Proceedings*, pages 789–795. JMLR.org. (Cited on page 19)

Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. (Cited on pages 8, 15, 54, 74, 76, and 79)

Tang, Y. and Eliasmith, C. (2010). Deep networks for robust visual recognition. In Fürnkranz, J. and Joachims, T., editors, *Proceedings of the 27th International Conference on Machine Learning, June 21-24, 2010, Haifa, Israel*, pages 1055–1062. Omnipress. (Cited on pages 28, 33, 40, and 41)

Tang, Y., Salakhutdinov, R., and Hinton, G. (2012). Robust boltzmann machines for recognition and denoising. In *IEEE Conference on Computer Vision and Pattern Recognition, 2012, Providence, Rhode Island, USA*. (Cited on pages 27 and 47)

Tapia, J. E. and Perez, C. A. (2013). Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of lbp, intensity, and shape. *IEEE Transactions on Information Forensics and Security*, 8(3):488–499. (Cited on pages 74 and 86)

Teh, Y. W. and Hinton, G. E. (2001). Rate-coded restricted boltzmann machines for face recognition. In Leen, T., Dietterich, T., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 908–914. MIT Press. (Cited on page 20)

Tieleman, T. (2008a). Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM New York, NY, USA. (Cited on page 19)

Tieleman, T. (2008b). Training Restricted Boltzmann Machines using Approximations to the Likelihood Gradient. In *Proceedings of the 25th international conference on Machine learning*, pages 1064–1071. ACM New York, NY, USA. (Cited on page 28)

Tompson, J., Goroshin, R., Jain, A., LeCun, Y., and Bregler, C. (2014). Efficient object localization using convolutional networks. *CoRR*. (Cited on page 8)

Turcsany, D. and Bargiela, A. (2014). Learning local receptive fields in deep belief networks for visual feature detection. In *Neural Information Processing - 21st International Conference, ICONIP 2014, Kuching, Malaysia, November 3-6, 2014. Proceedings, Part I*, pages 462–470. (Cited on page 29)

Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces. *Proceedings CVPR '91., IEEE Computer Society Conference on*, pages 586–591. (Cited on page 75)

Uetz, R. and Behnke, S. (2009). Locally-connected hierarchical neural networks for gpu-accelerated object recognition. In *Conference on Neural Information Processing Systems (NIPS 2009)*. Workshop on Large-Scale Machine Learning: Parallelism and Massive Dataset. (Cited on page 53)

Villegas, M. and Paredes, R. (2011). Dimensionality reduction by minimizing nearest-neighbor classification error. *Pattern Recognition Letters*, 32(4):633 – 639. (Cited on pages 75 and 76)

Villegas, M., Paredes, R., Juan, A., and Vidal, E. (2008). Face Verification on Color Images Using Local Features. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–6. IEEE Computer Society. (Cited on pages 51, 54, 55, and 57)

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.-A. (2008). Extracting and composing robust features with denoising autoencoders. Technical Report 1316, Département d'Informatique et Recherche Opérationnelle, Université de Montréal. (Cited on page 23)

Wan, L., Zeiler, M., Zhang, S., Cun, Y. L., and Fergus, R. (2013). Regularization of neural networks using dropconnect. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1058–1066. (Cited on pages 48, 53, 54, 66, 70, and 92)

Welling, M., Rosen-zvi, M., and Hinton, G. E. (2005). Exponential family harmoniums with an application to information retrieval. In Saul, L., Weiss, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 17*, pages 1481–1488. MIT Press. (Cited on pages 11, 19, and 23)

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University. (Cited on pages 7 and 11)

Yang, M., Zhang, L., Feng, X., and Zhang, D. (2011). Fisher discrimination dictionary learning for sparse representation. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 543–550. IEEE Computer Society. (Cited on page 46)

Yu, K. and Zhang, T. (2010). Improved local coordinate coding using local tangents. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*, pages 1215–1222. (Cited on page 66)

Zeiler, M. D. and Fergus, R. (2013). Visualizing and understanding convolutional networks. *CoRR*, abs/1311.2901. (Cited on pages 8 and 15)

Zeiler, M. D., Ranzato, M., Monga, R., Mao, M. Z., Yang, K., Viet Le, Q., Nguyen, P., Senior, A. W., Vanhoucke, V., Dean, J., and Hinton, G. E. (2013). On rectified linear units for speech processing. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3517–3521. (Cited on page 11)

Zou, H. and Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, 67:301–320. (Cited on pages 29 and 30)