UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

DEEP LEARNING ARCHITECTURES FOR COMPUTER VISION

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Carlos Roig Marí

In partial fulfilment

of the requirements for the degree in

Science and Technology of Telecommunications

Advisors: Elisa Sayrol & Ramon Morros

Barcelona, September 2016

Abstract

Deep learning has become part of many state-of-the-art systems in multiple disciplines (specially in computer vision and speech processing). In this thesis Convolutional Neural Networks are used to solve the problem of recognizing people in images, both for verification and identification.

Two different architectures, AlexNet and VGG19, both winners of the ILSVRC, have been fine-tuned and tested with four datasets: Labeled Faces in the Wild, FaceScrub, YouTubeFaces and Google UPC, a dataset generated at the UPC.

Finally, with the features extracted from these fine-tuned networks, some verifications algorithms have been tested including Support Vector Machines, Joint Bayesian and Advanced Joint Bayesian formulation. The results of this work show that an Area Under the Receiver Operating Characteristic curve of 99.6% can be obtained, close to the state-of-the-art performance.

Resum

L'aprenentatge profund s'ha convertit en una part importat de molts sistemes a l'estat de l'art de múltiples àmbits (especialment de la visió per computador i el processament de veu). A aquesta tesi s'utilitzen les Xarxes Neuronals Convolucionals per a resoldre el problema de reconèixer persones a imatges, tant per verificació com per identificació.

Dos arquitectures diferents, AlexNet i VGG19, les dues guanyadores del ILSVRC, han sigut afinades i provades amb quatre bases de dades: Labeled Faces in the Wild, FaceScrub, YouTubeFaces i Google UPC, un conjunt generat a la UPC.

Finalment, amb les característiques extretes de les xarxes afinades, s'han provat diferents algoritmes de verificació, incloent Màquines de Suport Vectorial, Joint Bayesian i Advanced Joint Bayesian. Els resultats d'aquest treball mostres que un Àrea Baix la Curva de la Característica Operativa del Receptor por arribar a ser del 99.6%, propera al valor de l'estat de l'art.

Resumen

El aprendizaje profundo se ha convertido en parte de muchos sistemas en el estado del arte de múltiples ámbitos (especialmente en visión por computador y procesamiento de voz). En esta tesis se utilizan las Redes Neuronales Convolucionales para resolver el problema de reconocer a personas en imágenes, tanto para verificación como para identificación.

Dos arquitecturas diferentes, AlexNet y VGG19, ambas ganadores del ILSVRC, han sido afinadas y probadas con cuatro conjuntos de datos: Labeled Faces in the Wild, FaceScrub, YouTubeFaces y Google UPC, un conjunto generado en la UPC.

Finalmente con las características extraídas de las redes afinadas, se han probado diferentes algoritmos de verificación, incluyendo Maquinas de Soporte Vectorial, Joint Bayesian y Advanced Joint Bayesian. Los resultados de este trabajo muestran que el Área Bajo la Curva de la Característica Operativa del Receptor puede llegar a ser del 99.6%, cercana al valor del estado del arte.

To my parents and my uncle “Padri” for supporting me all this years.

Acknowledgements

The author would like to thank the GPI, especially Albert Gil and Josep Pujal for allowing the use of the Development Platform of the UPC and their assistance with any kind of problem that could had happened.

Also thank the supervisors of the project, Elisa Sayrol and Ramon Morros, for giving me the chance to do this project, their guidance through it and finally their backing to get one step further.

Revision history and approval record

Revision	Date	Purpose
0	02/08/2016	Document creation
1	27/09/2016	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Carlos Roig Marí	roiginbag@gmail.com
Elisa Sayrol Clols	elisa.sayrol@upc.edu
Josep Ramon Morros Rubió	ramon.morros@upc.edu

Written by:		Reviewed and approved by:	
Date	25/09/2016	Date	27/09/2016
Name	Carlos Roig	Name	Elisa Sayrol & Ramon Morros
Position	Project Author	Position	Project Supervisors

Table of contents

List of Figures

List of Tables:

1. Introduction

1.1. Objectives

This project is carried out at the Signal Theory and Communications department (TSC) from the Universitat Politècnica de Catalunya (UPC).

The purpose of this project is to build a face recognition system for TV broadcast programmes, within the framework of a larger project that builds a multi-modal annotation system.

The project is implemented using state of the art techniques in the Machine Learning field known as Deep Learning. In particular, Convolutional Neural Networks are implemented using a library called Keras. Convolutional Neural Networks are used in computer vision applications for searching, understanding images, apps, medicine, self-driving cars and many other tasks. These networks will be trained and tested with labelled databases of face images.

The final objective is to implement a verification system, where given a pair of images, the system will decide if they belong to the same identity or not.

The project main goals are:

1. Learning and understanding the fundaments of Convolutional Neural Networks.
2. The implementation of a basic face recognition system.
3. Implementation of a very deep network.
4. The implementation of the face verification system.
5. Presenting results of the accuracy of the system.

1.2. Requirements and specifications

In order to execute and reproduce all the test and the implementations that are explained in this thesis, a computer with enough processing power or a server is required. In the case of this project the server used was the Development Platform from the Image Processing Group at the UPC. The server needs about 60 GB of accessible RAM to fit the deep neural network models and store the databases and a GPU (the newest possible NVidia card) to speed up the training stages.

The software required is Python 2.7 or 3.3+ and Keras with the following libraries NumPy which is a mathematics library, the Machine Learning library scikit-learn, Cuda 7.0 or higher with cuDNN to accelerate the process with the GPU and matplotlib for plotting the results.

1.3. Methods and procedures

This thesis follows the problem addressed in the Master Thesis done by Sergi Delgado in September of 2015 which was supervised by Elisa Sayrol and Ramon Morros, and done under the framework of the Camomile project, which is a collaborative project done by six participants from four European countries in which the UPC takes part. The Camomile project develops a collaborative annotation framework for multi-modal, multi-lingual and multi-media data. This project will be done under this framework, but will not be a part of the Camomile project, and will focus on the person discovery task proposed at the MediaEval 2015.

It is important to mention that despite not continuing the work done by Sergi Delgado, some results and interesting approaches from his thesis [1] were taken into account.

The project was proposed by the supervisors of the project, Elisa Sayrol, jointly with Ramon Morros.

1.4. Work plan

Work Packages

<table>
<tr><td>Project: Documentation</td><td colspan="2">WP ref: 1</td></tr>
<tr><td>Major constituent: Document</td><td colspan="2">Sheet 1 of 9</td></tr>
<tr><td rowspan="2">Short description: Study and understand the concepts of the Convolutional Neural Networks and the library TensorFlow that will be used in the development of the project. Also study the different databases that are available and prepare a list of features of each database.</td><td colspan="2">Planned start date:15/02/16
Planned end date: 09/03/16</td></tr>
<tr><td colspan="2">Start event:19/02/16
End event:16/03/16</td></tr>
<tr><td>Internal task T1: (1.1) Stanford deep learning course CS231n
Internal task T2: (1.2) TensorFlow Udacity Deep Learning course
Internal task T3: (1.3) Research on face databases
Databases: CelebFaces+, YouTube Faces, Megaface, Labeled Faces in the Wild, MediaEval 2015 test.
Internal task T4: (1.4) Very deep Convolutional Networks (VGG)
Paper: Very Deep Convolutional Networks for Large-Scale Image Recognition</td><td>Deliverables:</td><td>Dates:</td></tr>
</table>

<table>
<tr><td>Project: Project proposal and Work plan</td><td colspan="2">WP ref: 2</td></tr>
<tr><td>Major constituent: Document</td><td colspan="2">Sheet 2 of 9</td></tr>
<tr><td rowspan="2">Short description: First document to be delivered consisting on a project plan proposal and a work schedule that will be evaluated by the supervisors of the project.</td><td colspan="2">Planned start date: 27/02/16
Planned end date: 01/03/16</td></tr>
<tr><td colspan="2">Start event: 27/02/16
End event: 01/03/16</td></tr>
<tr><td></td><td>Deliverables:
Work plan</td><td>Dates:
01/03/16</td></tr>
</table>

Project: Basic verification system	WP ref: 3	
Major constituent: Algorithmic implementation	Sheet 3 of 9	
Short description: Basic implementation of a verification system, consisting in the usage of a pre-trained neural network, extracting the 4096 vector of features from the last fully connected layer, and compare the two images using different types of classifiers to decide if the identities are the same or not.	Planned start date: 10/03/16 Planned end date: 01/04/16	
	Start event:11/03/16 End event: 30/03/16	
Internal task T1: (3.1) Implementation Internal task T2: (3.2) Test with a threshold classifier Internal task T3: (3.3) Test with a naive Bayesian classifier Internal task T4: (3.4) Results	Deliverables:	Dates:

Project: Deeper network	WP ref: 4	
Major constituent: Algorithmic implementation	Sheet 4 of 9	
Short description: Implementation of a deeper network using an existing Very Deep Architecture, train and test it and finally present results.	Planned start date: 02/04/16 Planned end date: 12/05/16	
	Start event: 01/04/16 End event: 14/05/16	
Internal task T1: (4.1) Implementation Internal task T2: (4.2) Test and train Internal task T3: (4.3) Results	Deliverables:	Dates:

Project: Project critical review	WP ref: 5
Major constituent: Document	Sheet 5 of 9
Short description: Write the project critical review	Planned start date: 02/05/16 Planned end date: 09/05/16
	Start event: 05/05/16 End event: 09/05/16

	Deliverables: Project critical review	Dates: 09/05/16

Project: Face recognition architecture	WP ref: 6	
Major constituent: Algorithmic implementation	Sheet 6 of 9	
Short description: Adapt and propose an architecture of the network to improve performance and implement the deeper network into the face recognition system.	Planned start date: 25/04/16 Planned end date: 08/08/16	
	Start event: 10/05/16 End event: 15/08/16	
Internal task T1: (6.1) Implementation Internal task T2: (6.2) Test	Deliverables:	Dates:

Project: Results and analysis	WP ref: 7	
Major constituent: Algorithmic implementation	Sheet 7 of 9	
Short description: Test with databases and adjust the parameters of the network. Review the system implementation and check the performance of the system (Number of computations and parameters of the network). Comparison with the ImageNet basic system will be carried out.	Planned start date: 01/08/16 Planned end date: 31/08/16	
	Start event: 16/08/16 End event: 15/09/16	
Internal task T1: (7.1) Results and analysis	Deliverables:	Dates:

Project: Final report drafting	WP ref: 8
Major constituent: Document	Sheet 8 of 9
Short description: Prepare the final report of the project and review it.	Planned start date: 01/09/16 Planned end date: 27/09/16
	Start event: 01/09/16

	End event: 25/09/16	
Internal task T1: (8.1) Draft Internal task T2: (8.2) Revision	Deliverables:	Dates:

Project: Final report	WP ref: 9	
Major constituent: Document	Sheet 9 of 9	
Short description: Deliver the final report done in the WP8.	Planned start date: 27/09/16 Planned end date: 27/09/16	
	Start event: 27/09/16 End event: 27/09/16	
	Deliverables: Final report	Dates: 27/08/16

Table 1. Work packages.

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
2	5	Project Work plan	Milestone	3
5	14	Critical design review	Milestone	13
9	23	Final review	Milestone	18

Table 2. Milestones.

GANTT Diagram

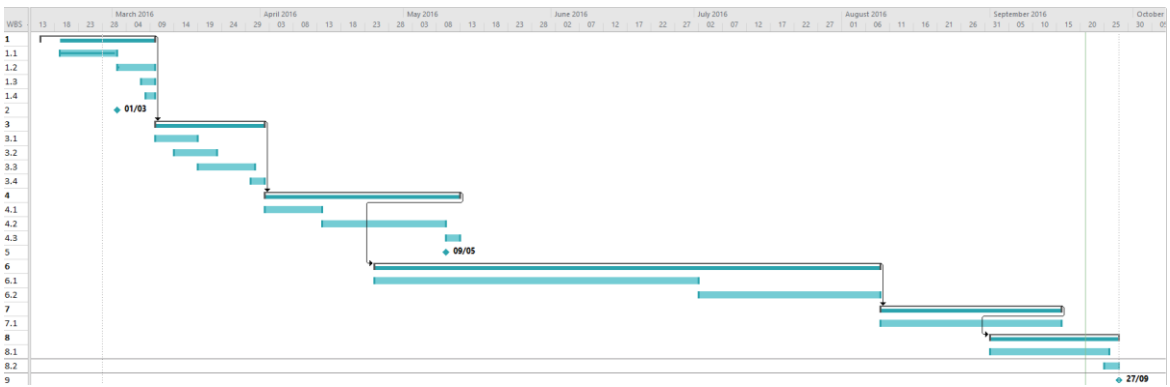


Figure 1. GANTT Diagram.

1.5. Deviations

One of the principal requirements was to use the library TensorFlow that was released to the public a few months before the start of this project. By the start of the project there were many different problems with this library and another system was chosen.

The other problem that caused the delay of the thesis was implementing the face verification algorithms Joint Bayesian and Advanced Joint Bayesian, these implementations were not working by different reasons such as errors in code, bad selection of images from the databases and other problems that ended in not being able to finish the project in the initially proposed schedule.

2. State of the art:

2.1. Deep Learning

Deep Learning is a branch of Machine Learning, concretely on the subfield of Artificial Neural Networks (ANN), based on a set of algorithms that attempt to model high-level abstractions in data by using a deep graph with multiple processing layers, composed of multiple linear and non-linear transformations. The first architectures were conceived in the nineties, but the computational cost was too high for the time and other techniques took the spotlight.

As stated in [2], Artificial Neural Networks were inspired by biological neural networks. An ANN consists on a structured organization of multiple artificial neurons. These neurons receive one or multiple inputs and emit an output based on the weighted sum of the inputs and a function. The function applied to the weighted sum is called the activation function.

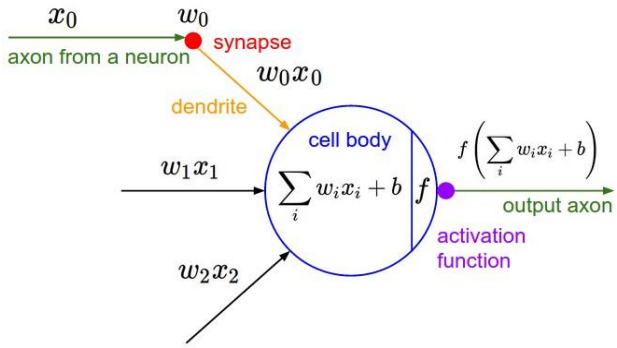


Figure 2. Representation of an artificial neuron compared to a biological neuron.

These neurons are grouped creating layers, by stacking these layers the networks are configured. The first layer of the network is known as the input layer, the intermediate are known as hidden layers and the last one is the output layer. The number of hidden layers will denote the depth of the network. The more layers the network has, the deeper the network will be.

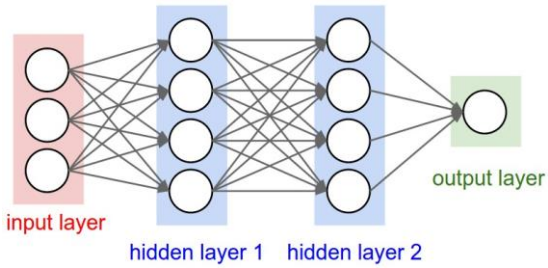


Figure 3. Example of a Neural Network with 3 layers.

The layers of the network perform different tasks based on the number of connections and their activation functions. Some of these types of layers are: the fully connected layers, which take into account all the neurons from the previous layer; the pooling layers, that perform a downsampling operation, for example, max pooling takes the maximum value from the inputs; or the convolutional layers that compute the output of neurons that are connected to a local region of the input. The networks based in this last kind of layers are known as Convolutional Neural Networks (CNN) or ConvNets.

The Convolutional Layers were created to deal with the amount of information in images. A 3 channel image contains lots of pixels that have some kind of information. But images are quite redundant regarding its content so using layers that connect every single pixel to many neurons were impossible to implement in terms of computation, that is why the convolutional layers were introduced, as mentioned before, these layers are connected to portions of the input and perform a convolution-like operation. The portions, called filters of kernels, usually have the shape of a square of size 3x3, 5x5 or 7x7 pixels, this shape is called the receptive field of the filter. These filters are passed by steps of 1, 2 or 3 pixels, this step is called the stride of the filters and by using a value higher than 1 the dimensionality of the output will be reduced. Finally if the input images does not fit with the filter's size and stride, a padding operation has to be performed. This padding is usually zero padding, which consist in adding zeros to the borders of the images to fit the required size, or a mirror padding that consist in mirroring the pixel values to the border.

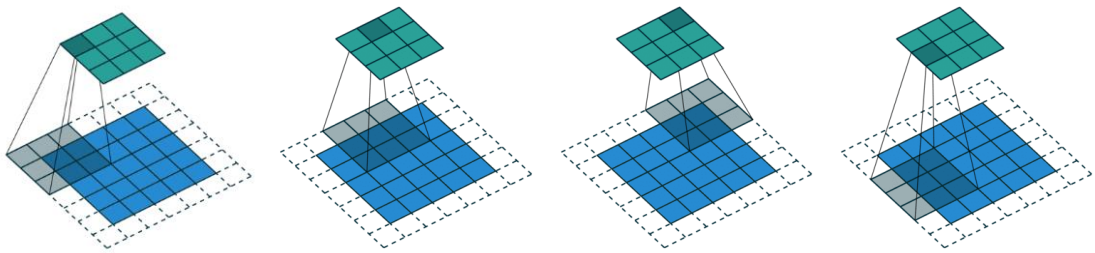


Figure 4. Example of a sequence of convolutional 3x3 filters with a stride of 2 and padding.

In 1998, the LeNet-5 architecture was proposed in [3], this structure was used to recognise handwritten and machine-printed characters and was one of the first architectures that used convolutional layers.

Some years later, in 2005, in [4] is explained the value of GPUs for machine learning to compute high volumes of operations faster than with a CPU. The usage of GPUs was refined the following years [5] and in 2012 deep neural networks significantly improved the best results of classification from many image databases, including the MNIST database, the CIFAR10 dataset and the ImageNet dataset.

The network that improved the results in the ImageNet dataset and also won de ILSVRC-2012 is known as AlexNet [6] [Figure 5]. This network takes as its input an image of 224x224 pixels with 3 colour channels. The number of layers is 8 (5 convolutional and 3 fully connected layers). The first convolutional layer filters the input image with 96 kernels of size 11x11x3 with a stride of 4 pixels. The second convolutional layer takes as input the response-normalized and max pooled output of the first convolutional layer and filters it with 256 kernels of size 5x5x48. The third convolutional layer has 384 kernels of size 3x3x256 connected to the normalized and max pooled outputs of the second layer. The fourth has 384 kernels of size 3x3x192, and the fifth has 256 kernels of size 3x3x192. The output of the fifth layer is connected to the first fully connected layer, with 4096 neurons. Then the first fully connected is connected to the second which also has 4096 neurons and the second is connected to the third, which has 1000 neurons corresponding to the ImageNet classes. Finally a softmax function is applied to last fully connected layer in order to get the class scores. The softmax function follows the equation:

$$\sigma(\boldsymbol{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \; for\; j = 1, \ldots, K, \tag{1}$$

where K is the dimensionality of the vector $\boldsymbol{z}$ and $\boldsymbol{\sigma}(\boldsymbol{z})$ is a vector in the range (0,1) that add up to 1.

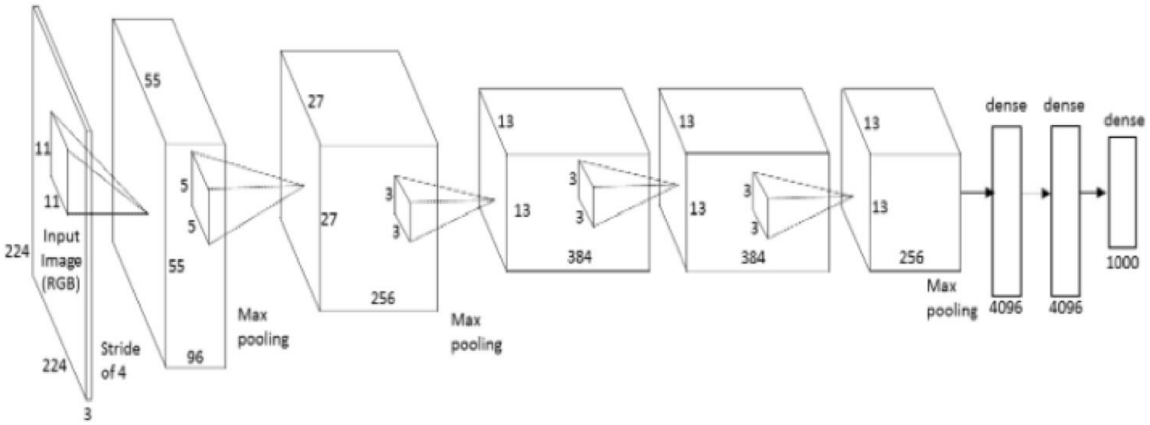


Figure 5. AlexNet network architecture.

The following years other architectures have been proposed with better results in the challenge, concretely 2014 VGG [7] won the challenge with a network that is widely used in many different applications, for example in [8]. Another architecture known as GoogLeNet [9] was proposed in the same year challenge, which implemented a new kind of layer named inception module.

The VGG architectures changed the receptive fields of the filters of all the layers to 3x3 filters with a stride of 1. VGG proposed in [7] six different architectures. The two deepest networks [Table 3. Columns D and E] are the ones with the best results. These architectures have 16 and 19 layers respectively being the last 3 layers, of each architecture, fully connected layers and the rest convolutional ones. The depth of the filters starts at 64 for the first two convolutional layers, then 128 for the next two, after that, the next 3[D]/4[E] layers have 256 kernels and the rest of the layers have 512 kernels. The fully connected layers have the same configuration as the ones in the AlexNet architecture two fully connected layers of 4096 neurons followed by a 1000 neuron layer.

All these architectures part from the idea of LeNet [3], where the input passes through some convolutional layers with a ReLU activation, which returns the maximum value between the input and 0. After the stack of convolutional layers, the output is flattened and it is connected to the fully connected layers that will compute the output of the network and with the classification layer a class will finally be selected.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Table 3. Different VGG models architectures.

2.2. Training

As stated in [10], training deep neural networks is known a hard task. In order to tune the weights of each neuron and its biases to achieve the best possible performance in the network, a careful training stage has to be executed.

There are some ways to do the training. Supervised learning consists in the task of inferring a function from labelled training data. This data consists of a set of training examples where the class of each example is known, so the training set will consist of a number of different input objects and the desired output value for each of these inputs. The training set, many times is divided into two subsets, one for the actual training, and another to control how the training is going, this subset is called the validation set.

Unsupervised learning is another way of training sets, but in this case, the classes of the data are not known beforehand. The objective of these methods is to form groups or clusters of data that are similar regarding some features, and assign to each cluster a label or class, which may not correspond with previously established classes. A good example is the K-Means Clustering algorithm [11], a method of vector quantization that makes a partition of n observations into k clusters, and assigns one cluster to each of the n observations.

There are other methods like [12] that combines both supervised and unsupervised learning methods, these are called semi-supervised methods. These methods consider some part of the unlabelled training set as labelled and starts generating clusters based on the made up labels.

In the process of training a network with supervised learning, the objective is to minimize a function that is called the loss or objective function. The network will be trained by updating

the weights of its layers after passing images through it and comparing the results with the desired output value, if both values match, the loss will decrement and, if the values do not match, the loss will increment. The number of times the whole training dataset is passed through the network, is determined by a parameter called epoch.

To retrain from scratch a deep neural network like VGG, the amount of images used [7] is above one million. In many different cases the dataset used is not big enough to meet this requirement, so two new techniques are proposed.

The first one is to fine-tune an already pre-trained network. By doing this, the network is adapted to solve a particular problem different from the original purpose of the network. The authors of [13] explain the benefits of using fine-tuning against other methods. The methodology to adapt a model consists in freezing the weights of some part of the network, typically the layers that are near the input layer, so when the training process starts, these layers are not updated and only the weights and biases from the desired layers are modified.

The other technique, which is also used in other application fields of ML, is called data augmentation, which consist in enlarging the size of the training dataset by performing different transformations over the data. The transformations that can be done are rotations, horizontal or vertical reflections, add white noise to the set, change of brightness, and many more. The transformations may vary from the dataset that is being used, for example, a hand-written dataset like MNIST would accept transformations like rotations or changing the brightness but reflecting horizontally or vertically the images would break the logic of the set. In [6] two types of data augmentation are implemented, the first one consisted in image translations and reflections that augmented the size of the set by 2048, and the second transformation consisted in the manipulation of the RGB colour channels.

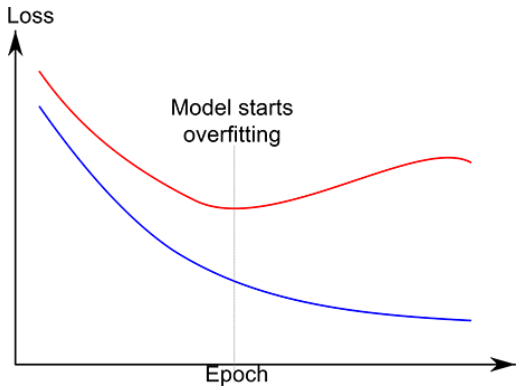


Figure 6. Example of Overfitting with the train loss in blue and validation loss in red.

Using data augmentation is also a method that could solve one of the main problems of the training stage which is overfitting. Overfitting is produced when the model overadapts to the training dataset. The loss produced by the network will keep decreasing as far the train stage goes, but the loss of the verification or test subsets will start to increment at some point [Figure 6]. This point where the verification loss is at the minimum is where the training should stop, but this point may not be the best possible solution for the addressed problem. By using data augmentation, the number of samples is increased so the point where the model starts to overfit is closer to the best possible verification loss result.

2.3. Classification and Verification

One of the typical tasks inside the field of Computer Vision (CV) is the task of recognition. In the field of recognition there are many different sub-tasks. In this thesis two of these

tasks are addressed, the first sub-tasks is known as classification (or identification) and the second is the task of verification.

The classification task, also called recognition (or identification) task, consist in giving one class or label from a previously defined set to an object. The set of labels that will be used in this thesis is a list of celebrities and the objects will be images of faces from the set. On the other hand, the verification task consists on deciding if two objects belong to the same class or not.

The classification is performed in the training stages of the process, where the network is fine-tuned to the new dataset and the weights of the last fully connected layers are adapted to maximize the difference between the different classes.

The verification task is performed using the outputs extracted from passing the images of the dataset through the model and extracting the values in the last layer before the classification layer. This layer is the layer that was fine-tuned in the training step and has adapted to the new dataset. With these outputs, many different techniques can be executed in order to achieve the best possible verification results.

Some of these are a threshold classifier based in the norm of the features, Support Vector Machines (SVM), or more recent techniques like Joint Bayesian [12] or the Advanced Joint Bayesian [13]. These advanced methods achieve better results in the task but have higher computational costs.

3. Methodology / project development:

3.1. System: Keras

The system that was initially chosen for this thesis was TensorFlow [14], which is an open source software library for machine learning written in Python and developed by the Google Brain Team. TensorFlow was recently released to the public and by the start of the thesis there were some compatibility issues on the UPC servers. So in order to proceed with the project a new system was selected.

Keras [14] is the system used in this thesis. It is a modular neural network library written in Python capable of running on top of either TensorFlow or Theano. The library was conceived to let the users start experimenting as fast as they could, being able to go from idea to result with the least possible delay.

The reasons why TF was selected and therefore Keras were that both systems are optimized to perform deep learning tasks. Both systems are implemented in Python which allow the user to work with them in a compact way without having to use multiple files. Both systems can run on top of both CPU and GPU which makes them very fast. And there is a huge community working with both of them and it is easy to find trained models or, for example, in the case of Keras, there is a library to convert trained models from Caffe, which is a deep learning framework developed by the Berkeley Vision and Learning Center [23] to Keras [15].

With Keras, the user first defines a model, which can be selected between a Sequential model or a Graph model. In a Sequential model, the layers are stacked and the output from a layer feeds the input of next layer until it reaches the output layer. In the other hand, the Graph model allows the users to get the output from a desired layer and feed that output to a desired layer, permitting the generation of multiple output networks or getting the output in an intermediate layer of the model.

Once the model is defined, it has to be compiled in order to start the training. The Keras “compile” function requires two parameters that need to be tweaked. These parameters are:

- Optimizer: This parameter will determine the learning and convergence of the model. There a lot of predefined optimizers in Keras, some of them are Stochastic Gradient Descend (SGD), Adam, RMSprop and Adagrad.
  All the optimizers have parameters that can also be modified, each optimizers has its own parameters, but there is one that is shared between all of them, the Learning Rate. This parameter will define how much the weights are updated after each epoch. For a high Learning Rate, the weight change will be higher than for a small Learning Rate, after each epoch. Also, another important parameter is the weight decay which is an additional term in the weight update rule that causes the weights to exponentially decay to zero, if no other update is scheduled.
- Loss: The second parameter, will define the objective of the training that the model has to optimize. There are many different objectives defined, for example: mean square error (MSE), mean squared logarithmic error (MSLE), categorical crossentropy that computes de logarithmic difference of the output with all the classes, and many more.
- Metrics: This parameter is optional and allows the user to see the accuracy of the model after each training step.

With the model compiled, the training can be started. The Keras function to train a model is called “fit” and has a lot of parameters that can be modified as well, some of them are the following:

- X: A Numpy array of the training data.
- Y: A Numpy array of the target data.
- Batch Size: The number of samples per gradient update.
- Number of epochs: The number of times to iterate over the training data.
- Callbacks: This parameter allows the user to save the weights of the network after each epoch if the loss is lower than any previous value.
- Validation Data: Is the sub-set that will validate how the model is performing.
- Shuffle: Allows the user to automatically shuffle the training data after each epoch.

3.2. Datasets

In this thesis, some of the most relevant datasets for face identification and verification where used alongside with some sets from the GPI server.

The first datasets that were tested are CelebA [24], MSRA-CFW [25] and YouTubeFaces (YTF) [17]. These datasets contain images from celebrities and were generated by getting images from the net, specially, YTF was generated with images from different frames of YouTube Videos. YTF is the only database with indications to crop the face of each frame from the previously mentioned datasets, so CelebA and MSRA-CFW were set aside.

The problem with YTF is that about a third of the different identities only have images from frames of the same video, and many of the videos are from interviews or TV shows where there is not a lot of movement, so the images are really similar.

Two databases that were in the GPI server were used to solve this similarity problem, which are Labelled Faces in the Wild (LFW) [18] and FaceScrub [19]. Like the sets mentioned before, these databases were generated from celebrity images collected from the internet. These sets have a problem with the image balance for some identities, in the case of LFW about a third of the identities only have one image available. Also mention that the images from LFW have the entire head of the identity and some background and the FaceScrub identities only have the face.

The last dataset that was used is called Google UPC, it was generated at the UPC by grabbing images of celebrities from Google.

Database	Number of identities	Number of images
LFW	5.749	13.233
FaceScrub	695	141.130
YTF	1.595	620.953
CelebA	10.177	202.599
MSRA-CFW	1583	202.792
Google UPC	293	19.708

Table 4. Comparison of all the datasets used in this thesis.

For this thesis some sub-sets based on the previously mentioned were generated, the exact sets used were the following three:

- Merged dataset: A merge of LFW and FaceScrub containing 546 identities and 21 images of each identity.

- YTF subset: This was generated using only images from YTF, the number of identities is 1595, as the original set, but the number of images for each identity was reduced to 15 and picked randomly for every identity.
- Google UPC: This set was used without modifying the number of images or identities, so it has 293 identities and a total of 19708 images.

3.3. Retraining of VGG and AlexNet

Performing a full training of the networks would have been too slow, an enormous database is needed and probably would ended with a worse result than fine-tuning, so performing a full training was discarded [1].

To fine-tune a model in Keras there is a parameter in every layer that needs to be adjusted, the parameter, called trainable, has to be set to false in the layers that the user does not want to update the weights in the training stage. By doing so the layers with the trainable parameter equal to false will be "frozen" (they will not change their value).

In both VGG19 and AlexNet models, the only layers that were not frozen were the last fully connected and the classification layers. The classification layer needs to be changed for every database since it has to classify the different identities of the set.

There is another important step that has to be performed before the training starts, it is the pre-processing of the images. The images that will fine-tune the network have to be adapted to the images that trained the model in the first place. For example, the first modification is subtracting the mean of the dataset that generated the weights that are being tuned. The other modification is changing the colour scheme to the one of the initial dataset (RGB or BGR).

With the pre-processing finished, the fine-tuning can start. The function "fit", mentioned before, is called and the training start with the parameters decided. The parameters selected for this thesis are the following:

- Optimizer: SGD was the optimizer used because of its convergence rate and the different parameters that could be adjusted, this being the learning rate that was set up to $1e^{-4}$ and the weight decay to $1e^{-6}$.
- Loss: The objective decided for this task has to be the categorical crossentropy since is the most accurate parameter for performing a classification with multiple classes. Mathematically, this function computes $H(p,q) = -\sum_x p(x)\log\big(q(x)\big)$, where $H(p,q)$ is the crossentropy between p and q and p and q are two probability distributions.
- Regarding the training parameters, the data and objectives were images and labels of the dataset to which the network was being tuned, the batch size was set to 16(VGG19) and 32(AlexNet) since a bigger one could not fit in the available GPUs, in some tests by increasing the batch size the training sped up or even improved the accuracy of the network. The number of epochs that was selected was 200, but by using a callback to save the weights after each epoch, the training could manually be stopped when a point of convergence was reached. Finally the sets were splitted in train and validation according to [Table 5].

Finally data augmentation was tested in some network configurations and datasets to test its viability. Keras's data augmentation generates images with small changes based on the parameters desired by the user. These modified images are generated and changed randomly after each epoch. The different parameters that can be added with Keras are

subtracting the mean of the dataset or the mean of the sample, normalize with the standard deviation of the dataset or the current image, applying ZCA whitening, randomly rotate images for a maximum amount of degrees, shifting the images vertically or horizontally, zooming into the images and many other options.

Dataset	Train images	Validation images
Merged dataset	10466	1000
YTF subset	18925	5000
Google UPC	15708	4000

Table 5. Train/Validation images splits for the selected databases.

3.4. Verification implementations

With the networks fine-tuned, the last task of this project is addressed. In this thesis many different verification techniques have been tested. In order to perform the task of verification, the output of the classifying layer is not enough and more values are needed. To have more information than the one obtained from the classification layer, this last layer is removed. The resulting is an array of 4096 parameters that will be called the feature array or features.

The first and most simple test to implement was a threshold classifier, which consisted in getting a pair of features, subtract one vector from the other and compute its norm. This was done for 20000 pairs and finally with a partition of these, the threshold was trained and tested with the other partition.

The following was to train a Support Vector Machine (SVM), which are a set of supervised learning methods that consist in generating a multidimensional partition of the space of the features by feeding a subset of them to train the SVM. The threshold between classes can be modelled by tweaking the kernel of the SVM, for example, with a linear kernel the threshold between classes will be a line and with a RBF kernel the threshold between classes will follow a quadratic function. In the thesis the linear SVM was the first to be implemented and then the RBF was tested which parameters where tuned for the best possible performance. The parameters of the RBF are gamma and C, gamma defines how far the influence of a single training examples reaches and C trades off misclassification of training examples against simplicity of the decision surface.

Once these more classic methods were working properly, some state-of-the-art methods were tested. The first one was the Joint Bayesian (JB) classifier [12]. This method parts from the naive formulation where the faces $\{x_1, x_2\}$ are modelled as Gaussians. These probabilities are computed based on the hypothesis that the faces are from the same identity Eqn. (2) or from different identities Eqn. (3).

$$P(x_1, x_2 | H_I) = N(0, \Sigma_I) \quad (2)$$

$$P(x_1, x_2 | H_E) = N(0, \Sigma_E) \quad (3)$$

where Σ_I and Σ_E are two covariance matrices that can be estimated from the intra-personal pairs and extra-personal pairs respectively.

JB introduces a prior on the face representation which consist in the sum of two independent Gaussian variables that follow the equation:

$$x = \mu + \varepsilon, \quad (4)$$

where x is the observed face with the mean of all the faces subtracted, μ represents its identity, ε is the face variation within the same identity. The distribution of these variables follows:

$$\mu \sim N(0, S_\mu)$$

and

$$\varepsilon \sim N(0, S_\varepsilon),$$

where S_μ and S_ε are two unknown covariance matrices. Given the linear form of Eqn. (4) and the independent assumption between μ and ε, the covariance of two faces is:

$$\boldsymbol{cov}(x_1, x_2) = \boldsymbol{cov}(\mu_1, \mu_2) + \boldsymbol{cov}(\varepsilon_1, \varepsilon_2). \quad (5)$$

The covariance matrix of the probability distributions $P(x_1, x_2 | H_I)$ and $P(x_1, x_2 | H_E)$ can be derived as:

$$\Sigma_I = \begin{bmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{bmatrix}, \ \Sigma_E = \begin{bmatrix} S_\mu + S_\varepsilon & 0 \\ 0 & S_\mu + S_\varepsilon \end{bmatrix}. \quad (6)$$

With the above conditional joint probabilities, a ratio of log likelihood can be obtained:

$$r(x_1, x_2) = \log \frac{P(x_1, x_2 | H_I)}{P(x_1, x_2 | H_E)} = x_1^T A x_1 + x_2^T A x_2 - 2 x_1^T G x_2 \quad (7)$$

where

$$A = (S_\mu + S_\varepsilon)^{-1} - (F + G) \quad (8)$$

$$\begin{pmatrix} F + G & G \\ G & F + G \end{pmatrix} = \begin{pmatrix} S_\mu + S_\varepsilon & S_\mu \\ S_\mu & S_\mu + S_\varepsilon \end{pmatrix}^{-1} \quad (9)$$

Since S_μ and S_ε are two unknown covariance matrices that need to be learned from the data, an EM-like algorithm to jointly estimate them is developed.

In the E-step, for each subject with n images, the relationship between the variables $\boldsymbol{h} = [\mu; \varepsilon_1; \dots; \varepsilon_n]$ and the observations $\boldsymbol{x} = [x_1; \dots; x_n]$ is $\boldsymbol{x} = \boldsymbol{P}\boldsymbol{h}$, where $\boldsymbol{P} = [\boldsymbol{I} \mid diag(\boldsymbol{I})]$. The covariance matrix of $\boldsymbol{h}$ is $\Sigma_h = diag(S_\mu, S_\varepsilon, \dots, S_\varepsilon)$, being its distribution $\boldsymbol{h} \sim N(0, S_\mu)$. Therefore, the distribution of **x** is $N(0, \Sigma_x)$ with $\Sigma_x = \begin{bmatrix} S_\mu + S_\varepsilon & S_\mu & \dots & S_\mu \\ S_\mu & S_\mu + S_\varepsilon & \dots & S_\mu \\ \vdots & \vdots & \ddots & \vdots \\ S_\mu & S_\mu & \dots & S_\mu + S_\varepsilon \end{bmatrix}$.

The expectation of the variable $\boldsymbol{h}$ is computed as:

$$E(\boldsymbol{h}|\boldsymbol{x}) = \Sigma_h \boldsymbol{P}^T \Sigma_x^{-1} \boldsymbol{x}. \quad (10)$$

With the new estimated values of μ and ε from $\boldsymbol{h}$. The M-step updates the covariance matrices $S_\mu = cov(\mu)$ and $S_\varepsilon = cov(\varepsilon)$ with the estimated variables in the E-step.

Finally, with the covariance matrices, the log likelihood ratio is computed for pairs of images, with this ratio, a threshold can be trained and it will decide if the pair of images belong to the same subject or not.

When the 4096 features array was tested with Joint Bayesian, the time to perform a full convergence of the algorithm took more than three days. This time had to be reduced in

order to perform other tests. The approach taken to reduce this time was to reduce the dimensionality of the features.

In order to reduce the dimensionality two methods were tested, first a Principal Component Analysis (PCA) was performed, this procedure uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of linearly uncorrelated variables, the principal components. Then the number of components is selected based on user specifications.

The other method used is based in deep neural networks and is called autoencoder [20]. An autoencoder is a stack of layers that decreases the dimensionality of the input vector progressively. In this thesis three different autoencoders were used:

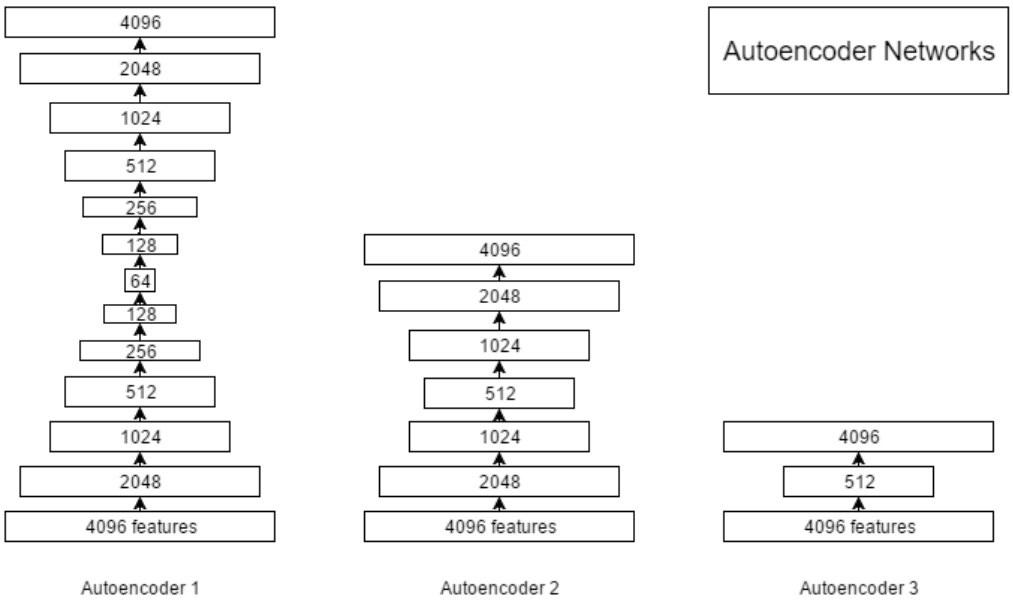


Figure 7. Autoencoder networks used in this thesis.

Using Autoencoders is better than just adding a fully connected layer before the classification layer in the network because they allow the user to extract the features in many different dimensions and the computational cost of training these autoencoders is way less than fine-tuning a network like VGG with an additional fully connected layer.

Finally, the last method tested for face verification was the Advance Joint Bayesian [13], the authors of the paper state that the Joint Bayesian method is imperfect and propose a corrected formulation.

The first correction that is proposed is to estimate the parameter μ in an iterative way rather than being fixed a priori. The second and more important change is that the JB computes the expectation of the variables directly, while a standard EM algorithm would compute the expectation from the log-likelihood of the variables.

Under the Gaussian assumption, AJB defines $x = h + \epsilon$ with its probability distributions $P(h) = N_{\mu,\Sigma_b}(h)$ and $P(\epsilon) = N_{0,\Sigma_\omega}(\epsilon)$, where μ, Σ_b and Σ_ω are model parameters.

In a standard EM algorithm the parameters of the model should be:

$$\left[\mu^{(t)}, {\Sigma_b}^{(t)}, {\Sigma_\omega}^{(t)}\right] = \underset{\mu,\Sigma_b,\Sigma_\omega}{\operatorname{argmax}} E_{\boldsymbol{h}|\boldsymbol{x},\mu^{(t-1)},{\Sigma_b}^{(t-1)},{\Sigma_\omega}^{(t-1)}} \times \ln P(\boldsymbol{x}, \boldsymbol{h} \mid \mu, \Sigma_b, \Sigma_\omega)$$

$$= \underset{\mu,\Sigma_b,\Sigma_\omega}{\operatorname{argmax}} E_{\boldsymbol{h}|\boldsymbol{x},\mu^{(t-1)},\Sigma_b{}^{(t-1)},\Sigma_\omega{}^{(t-1)}} \times \ln(P(\boldsymbol{h} \mid \mu, \Sigma_b, \Sigma_\omega) P(\boldsymbol{x} \mid \boldsymbol{h}, \mu, \Sigma_b, \Sigma_\omega)) \quad (11)$$

$$= \underset{\mu,\Sigma_b,\Sigma_\omega}{\operatorname{argmax}} E_{\boldsymbol{h}|\boldsymbol{x},\mu^{(t-1)},\Sigma_b{}^{(t-1)},\Sigma_\omega{}^{(t-1)}} \times \ln P(\boldsymbol{h} \mid \mu, \Sigma_b) P(\boldsymbol{\epsilon} \mid \Sigma_\omega)$$

where (t) is the (t)th iteration of the algorithm. By expanding the term in the first equation of (10):

$$E\left(\ln\left(\prod_{i=1}^{c} N_{\mu,\Sigma_b}(h_i)\right)\right) = \sum_{i=1}^{c} E\left(\ln N_{\mu,\Sigma_b}(h_i)\right) = -\frac{c}{2}\ln|\Sigma_b| - \quad (12)$$

$$-\frac{1}{2}\sum_{i=1}^{c} E\left((h_i-\mu)^T \Sigma_b{}^{-1}(h_i-\mu)\right) + const.$$

where c is the number of subjects. Then, maximizing (11) with respect to μ and Σ_b:

$$\mu^{(t)} = \frac{1}{c}\sum_{i=1}^{c} E\, h_i = \frac{1}{c}\sum_{i=1}^{c} \hat{h}_i^{(t)}, \quad (13)$$

$$\Sigma_b{}^{(t)} = \frac{1}{c}\sum_{i=1}^{c} E\left((h_i-\mu^{(t)})(h_i-\mu^{(t)})^T\right) = \frac{1}{c}\sum_{i=1}^{c}\left(cov(h_i) + (Eh_i-\mu^{(t)})(Eh_i-\mu^{(t)})^T\right)$$

$$= \frac{1}{c}\sum_{i=1}^{c}\left(\left(\hat{h}_i^{(t)}-\mu^{(t)}\right)^2 + S_i^{(t)}\right), \quad (14)$$

where $\hat{h}_i^{(t)} = \mu^{(t-1)} + \Sigma_b{}^{(t-1)}\left(\Sigma_b{}^{(t-1)} + \frac{\Sigma_\omega{}^{(t-1)}}{n_i}\right)(\bar{x}_i - \mu^{(t-1)})$ and $S_i^{(t)} = cov(h_i) = \Sigma_b{}^{(t-1)}\left(n_i\Sigma_b{}^{(t-1)} + \Sigma_\omega{}^{(t-1)}\right)^{-1}\Sigma_\omega{}^{(t-1)}$ with n_i equal to the number of samples for the subject i.

The last model parameter follows the equation:

$$\Sigma_\omega{}^{(t)} = \frac{1}{\sum_{i=1}^{c} n_i}\sum_{i=1}^{c}\sum_{j=1}^{n_i}\left(\left(\hat{\epsilon}_{ij}^{(t)}\right)^2 + S_i^{(t)}\right), \quad (15)$$

where $\hat{\epsilon}_{ij}^{(t)} = x_{ij} - \hat{h}_i^{(t)}$ with $i \in (1, c)$ and j $\in (1, n_i)$.

AJB adds the $S_i^{(t)}$ parameter during the update of Σ_b and Σ_ω, which is the main difference from the JB method and guarantees that the AJB will converge to the local maximum of the likelihood function. Which is defined as:

$$L(x_1, x_2) \sim s^T\Sigma_1 s - \Delta^T\Sigma_2\Delta, \quad (16)$$

where

$$s = x_1 + x_2 - 2\mu,$$

$$\Delta = x_1 - x_2,$$

$$\Sigma_1 = (\Sigma_\omega + \Sigma_b)^{-1} - (\Sigma_\omega + 2\Sigma_b)^{-1},$$

$$\Sigma_2 = \Sigma_\omega{}^{-1} - (\Sigma_\omega + \Sigma_b)^{-1}.$$

In this thesis the formulation above is implemented in Python and tested against the other classifiers.

4. Results

4.1. Fine-tuning of VGG19 and AlexNet

- VGG19 and AlexNet comparison

VGG19 and AlexNet were compared using the merged dataset. Also data augmentation (DA) was tested with both networks. It can be observed in [Figure 11] that AlexNet converges faster than VGG, which is because AlexNet is shallower, but the best result is obtained with VGG19.

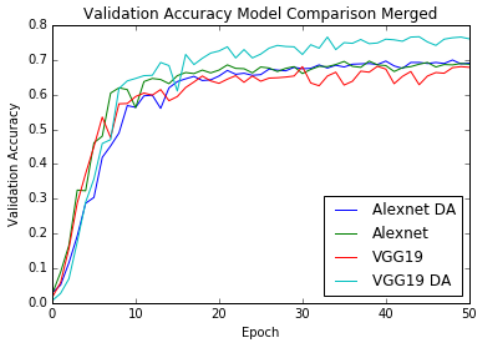


Figure 8. Validation Accuracy comparing AlexNet and VGG19 with the merged dataset.

From now on, the only network used was VGG19.

- YTF dataset with VGG19

The network was trained for 30 epochs and the accuracy value obtained is above 90%.

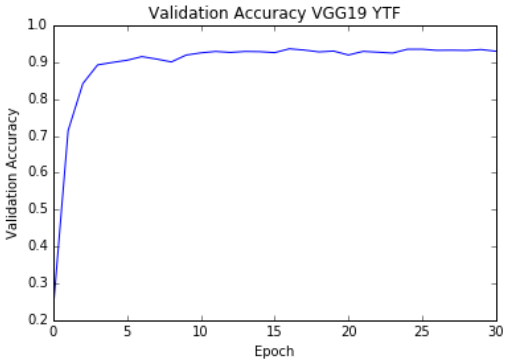


Figure 9. Validation Accuracy with VGG19 for the YTF subset.

The YTF dataset, as stated in 3.2, is composed of images from frames of YouTube videos which almost have no movement so the images of the faces are really similar. To the point that could be considered the same image with some kind of data augmentation. Also mention that for this dataset using data augmentation did not improve the accuracy of the network.

- Merged dataset with VGG19

The results for the merged dataset show that using DA in the last fully connected layer make improve the accuracy of the network by 10%, while fine-tuning the last two fully connected layers results in only an improvement of 5%.

Fine-tuning tests	Accuracy
Last fully connected layer no DA	70%
Last fully connected layer with DA	80%
Last 2 fully connected layers with DA	75%

Table 6. Accuracy comparison for VGG19 with the merged dataset.

The images from this dataset, that are being labelled incorrectly, have a pattern that can be easily seen, almost every incorrect image has the face sided. Which is known to be one difficult problem for the task of face classification.

- Google UPC with VGG19

The same result that has been seen in the merged dataset is found here. The Google UPC dataset has 293 identities and more than 19000 images that were extracted from Google and do not have a pattern like YouTubeFaces, so when data augmentation is applied, the results improve slightly.

Fine-tuning tests	Accuracy
Last fully connected layer no DA	82%
Last fully connected layer with DA	88%

Table 7 Accuracy comparison for VGG19 with Google UPC.

4.2. Classical classifiers for verification and dimensionality reduction

The performance of the verification is measured by the parameter AUC (Area Under the Curve) from the ROC (Receiver Operating Characteristic) curve.

Given a threshold parameter T, the pair of images is classified as "positive" if $X > T$, and "negative" otherwise. X follows a probability density $f_1(x)$ if the pair actually belongs to the class "positive", and $f_0(x)$ if not. Therefore, the True Positive Rate is given by $TPR(T) = \int_T^\infty f_1(x)dx$ and the False Positive Rate is given by $FPR(T) = \int_T^\infty f_0(x)dx$. The ROC curve plots parametrically $TPR(T)$ versus $FPR(T)$ with T as the varying parameter.

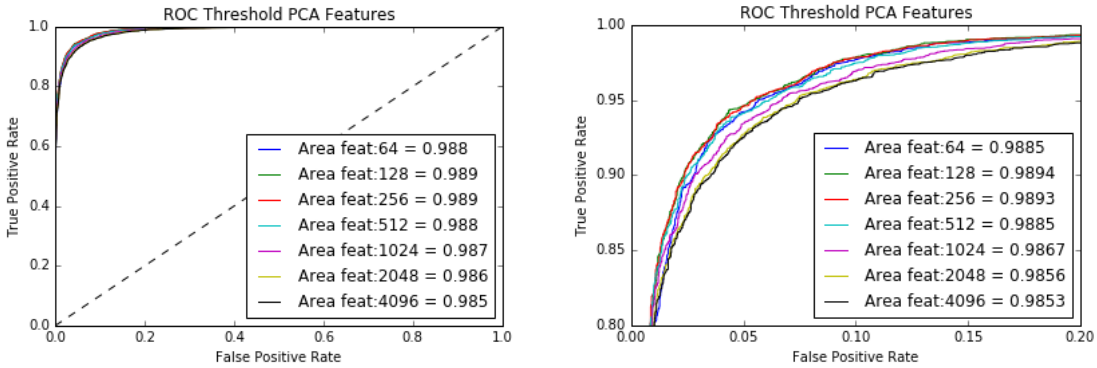
The AUC follows the equation:

$$A = \int_{\infty}^{-\infty} TPR(T)FPR'(T)dT. \tag{17}$$

It is important to mention that all the verification methods have been tested with the merged dataset. Extracting the features from the VGG19 network with the weights obtained by retraining the last fully connected layer with data augmentation.

- Threshold classifier

The results obtained with the threshold classifier are better than what was expected. The AUC is almost 99%, being 100% the best possible that would correspond to a perfect system. The results also show that the number of dimensional features do not make a big difference for this classifier.



Figures 10 & 11. ROC of the threshold classifier with PCA of the merged dataset (left) and with zoom (right).

The Autoencoders were also tested with the threshold classifier, giving the results shown in [Table 8].

Method	AUC
PCA 512 features	0.988
Autoencoder 1	0.858
Autoencoder 2	0.807
Autoencoder 3	0.869

Table 8. AUC comparison between autoencoders and PCA.

The autoencoder that gives the best result is the number 3 which is the one that converged slower in the training stage. But the results of the PCA are much better. In the next experiments the features used will only be the ones obtained with PCA.

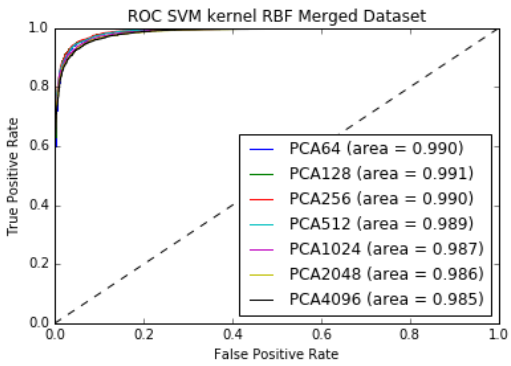
- SVM

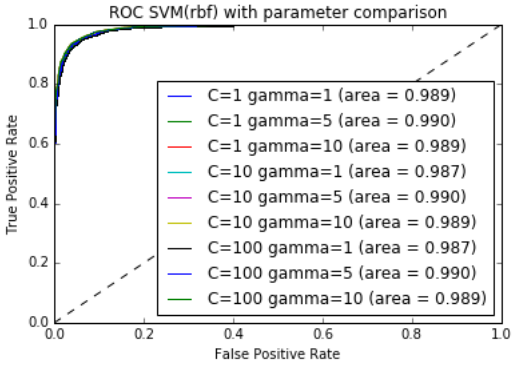
The first SVM configuration tested was with a linear SVM which gave a result worse than a random guess, the AUC is under 50% and this kind of SVM was discarded.

Number of PCA features	AUC
64	0.488
128	0.488
256	0.484
512	0.495
1024	0.479
2048	0.475
4096	0.474

Table 9. AUC comparison of the SVM with linear kernel.

The next test was with the SVM with a RBF kernel, which improved the results of the threshold classifier by a small percentage. As in the case of the threshold classifier, the PCA does not make a big improvement, but it is true that the smaller dimensions perform better than the bigger ones. To achieve this improvement the parameters C and gamma were tuned, with gamma equal to 5 being the best result for any value of C.

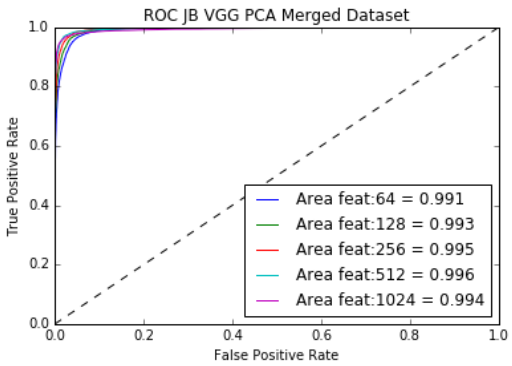


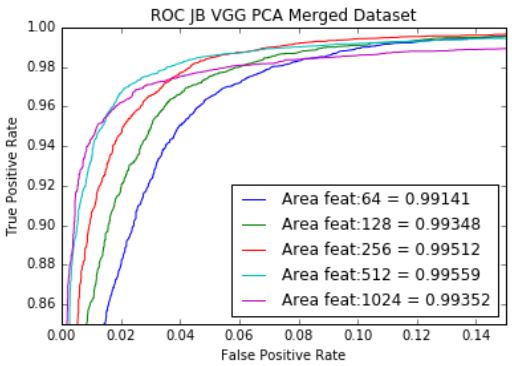


Figures 12 & 13. SVM with kernel RBF and PCA (left). Parameter comparison for SVM with kernel RBF (right).

4.3. Joint Bayesian

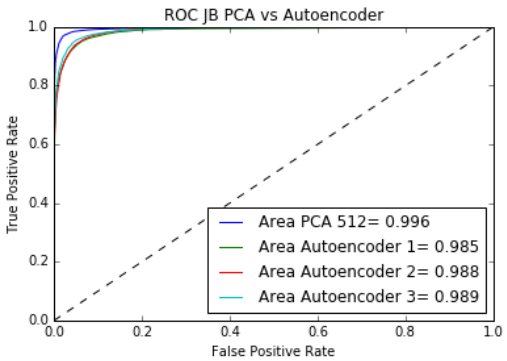
Joint Bayesian has given the best results for the verification task out of all the algorithms, the results improve slightly the ones obtained with the SVM but in this case the best dimension is PCA 512 instead of 128/256. It is also important to notice that the results only show values of the PCA until the 1024 features, which is because computing the PCA for 2048 and 4096 features lasted too much.

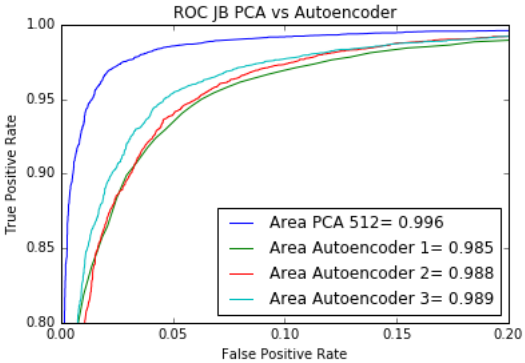




Figures 14 & 15. JB for the Merged Dataset with PCA (left) and zoomed image (right).

With the best classifier, the autoencoder values were, again, tested, giving a better result than for the threshold classifier but not reaching the results of the PCA. Also mention, the third autoencoder performs better than the other two, as in the threshold method.





Figures 16 & 17. Comparison of PCA and Autoencoder with the JB (left) and zoomed image (right).

4.4. Advanced Joint Bayesian

The algorithm implemented for the AJB following the pseudocode in [13] ended giving bad results. This method was supposed to converge faster than the JB and giving a result similar or better than the JB.

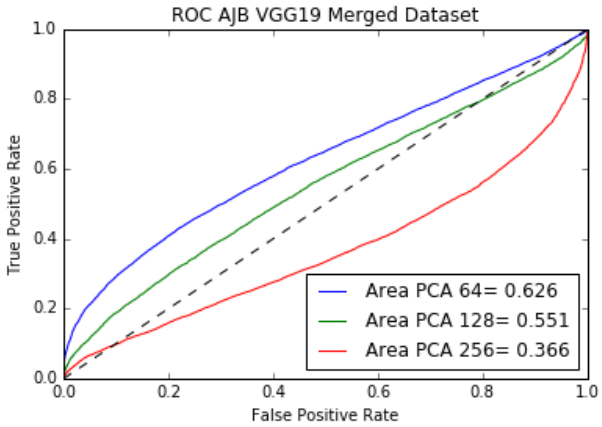


Figure 18. ROC AJB with VGG19 PCA features.

The results shown in the figure are too distant from the ones obtained with the JB, in order to find what could be the problem, the author of the AJB was contacted via email.

The author of the AJB stated that the AJB works better with Gaussian models of data and with the case of Convolutional Neural Networks, the Joint Bayesian outperforms the AJB.

Finally the time to compute each iteration of the AJB is higher than the time to compute an iteration for the JB with the same amount of features.

Features	JB	AJB
64	0.4s	2s
128	0.6s	6s
256	2.5s	32s
512	10s	180s
1024	60s	1320s
2048	300s	11880s

Table 10. Time required to compute one iteration of the JB and the AJB.

For the purpose of this project, the Joint Bayesian method is superior in every single aspect respect the Advanced Joint Bayesian.

4.5. Conclusions

First, the VGG19 model performs better than AlexNet, as it was expected, since VGG was published after AlexNet as an improved model for the task of classification, proving that its performance was better.

Then the datasets that were used to train the VGG19 perform different based on the type of images that generated the dataset. YTF has given the best results because the images have very little variations, since the faces are extracted from videos, while the merged dataset and Google UPC have faces from different pictures. That is also why for YTF DA does not work properly and for the merged dataset and Google UPC, DA improves the accuracy of the network.

Another interesting result is that the convolutional features extracted from the models have better results when being dimensionally reduced with PCA than with Autoencoders.

Finally, regarding the verification task, JB is the method that performs the best out of all the ones tested. AJB was supposed to work even better, but the results showed that it was not working properly.

The threshold classifier has proven to give very good results despite its simplicity, resulting in an AUC of only 0.66% less than the JB.

The final comparison of the best results is:

Method	AUC
Threshold classifier with PCA 128	0.9894
SVM kernel RBF with PCA 128	0.991
Joint Bayesian with PCA 512	0.996

Table 11. Comparison of the best methods used in this thesis.

5. Budget

This project was carried out using the personal computer of the author and the Development Platform server from the GPI of the UPC where all the experiments and tests were executed.
All the software used is open source or under a free to use for non-commercial profit license.

This project was started in February of 2016 and was ended in September 2016, a total of 7 months. Working 20 hours each week and considering that the salary of a junior engineer is around 1850€ a month, working full time (40h/week) would cost 6475€.

Two supervisors have assisted in the realization of this project. Each week, a 1 hour meeting was realized with both of the supervisors to check on the process of the project. The supervisors fee is 55€/h. A total of 32 meetings have been done in this project, with a total cost of 3520€.

Finally, if the server was outsourced to a company like Amazon, which offers access to their servers, the price for a server with enough RAM and a GPU is 2,7522€/hour. One week of this service would cost 462,88€ and for the total duration of this project, the cost of the server would have been 14812,16€.

Name	Cost/week	Number of weeks	Cost
Junior Engineer	202,34375€	32	6475€
Supervisors	110€	32	3520€
Outsourced Server	462,88€	32	14812,16€
Total			24807,16€

Table 12. Cost of the project.

The total cost of this project would be 24807,16€.

6. Conclusions and future development

The field of deep learning is improving every day, with new architectures, approaches and new interesting concepts which make the possibilities to extend this work very wide.

In this thesis it is shown that by using convolutional features extracted from very well-known models, the results obtained can be really good. It is important to decide the best dataset for the task addressed at any time. For example to test the face verification algorithms the YTF dataset achieve better results than the merged dataset because of the fact that the images are all from the same clips of video, which makes the dataset invalid for a proper test.

In the task of classification, new network architectures with recursive layers have become popular by winning the ILSVRC in 2015. With the improvements on the field and the publication of software like TensorFlow, the field will evolve even faster in the following years. Testing these new networks could be an interesting test for future projects, which would consist in implement the Inception-v3 [21] or Inception-v4 [22] models in TF and compare them with older models.

Finally, regarding the verification task, generative Bayesian methods like Joint Bayesian have achieved the best results in this task the last years and some improvements like the Advanced Joint Bayesian are published reporting better results than the Joint Bayesian. The theory behind them is interesting to study but for the case of AJB, in this thesis is shown that does not improve the results of JB for face verification with deep neural features.

References

[1] S. Delgado. "Deep Learning for Face Recognition". M.S. thesis, Escola Tècnica Superior d'Enginyeria de Telecomunicació, Universitat Politècnica de Catalunya, Barcelona, Spain, 2015.

[2] V. Sharma, S. Rai, A. Dev. "A Comprehensive Study of Artificial Neural Networks". *In International Journal of Advanced Research in Computer Science and Software Enginerring, IJARCSSE,* 2012

[3] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner. "Gradient-Based Learning Applied to Document Recognition" *In Proceeding of the IEEE, vol. 86, no. 11, pp 2278-2324*, August 2002. DOI: 10.1109/5.726791

[4] D. Steinkrau, P. Y. Simard, I. Buck. "Using GPUs for Machine Learning Algorithms". *In Proceedings of the Eighth International Conference on Document Analysis and Recognition* (ICDAR '05). IEEE Computer Society, Washington, DC, USA, 1115-1119. DOI=http://dx.doi.org/10.1109/ICDAR.2005.251

[5] D. Ciresan, U. Meier, J. Masci, L. Gambardella, J. Schmidhuber. "Flexible, High Performance Convolutional Neural Networks for Image Classification". *In Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence IJCAI, pp. 1237–1242,* Galleria 2, 6928 Manno-Lugano, Switzerland 2011.

[6] A. Krizhevsky, I. Sutskever, G. Hinton. "Imagenet classification with deep convolutional neural networks". *In Advances in Neural Information Processing Systems 25, pages 1106–1114*, 2012.

[7] K. Simonyan, A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". arXiv:1409.1556v6 [cs.CV] 10 Apr 2015

[8] L. A. Gatys, A. S. Ecker, M. Bethge. "A Neural Algorithm of Artistic Style". arXiv:1508.06576v2 [cs.CV] 2 Sep 2015

[9] C. Szegedy, W. Liu, Y. Jia et al. "Going deeper with convolutions". arXiv:1409.4842v1 [cs.CV] 17 Sep 2014

[10] H. Larochelle, Y. Bengio, J. Louradour, P. Lambin. "Exploring Strategies for Training Deep Neural Networks". *In Journal of Machine Learning Research 1 (2009) 1-40*. January 2009, Montréal, Québec, Canada.

[11] T. Kanungo, et al. "An Efficient k-Means Clustering Algorithm: Analysis and Implementation". *In IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 7 pp. 881-892, July 2002. DOI: 10.1109/TPAMI.2002.1017616

[12] D. Chen, X. Cao, L. Wang, F. Wen, J. Sun, "Bayesian face revisited: A joint formulation". In Proc. ECCV, 2012.

[13] Y. Liang, X. Ding, J. Xue. "Advanced Joint Bayesian Method for Face Verification". *In IEEE Transactions on Information Forensics and Security,* vol. 10, no. 2, pp. 346-354, December 2014. DOI: 10.1109/TIFS.2014.2375552

[14] M. Abadi, A. Agarwal, P. Barham et al. "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems". Google Research, 2015

[15] F. Chollet et al. "Keras: Deep Learning library for Theano and TensorFlow". [Online] Available: https://keras.io/. [Accessed: 23 September 2016].

[16] M. Bolaños et al. "Caffe to Keras conversion module" [Online] Available: https://github.com/MarcBS/keras. [Accessed: 23 September 2016].

[17] L. Wolf, T. Hassner, I. Maoz. "Face Recognition in Unconstrained Videos with Matched Background Similarity" *In IEEE Computer Vision and Pattern Recognition (CVPR)*, June 2011. DOI: 10.1109/CVPR.2011.5995566

[18] E. Learned-Miller, G. Huang, A. RoyChowdhury, H. Li, G. Hua. "Labeled Faces in the Wild: A Survey" [Online] Available: https://people.cs.umass.edu/~elm/papers/LFW_survey.pdf. [Accessed: 23 September 2016].

[19] H. Ng, S. Winkler. "A Data-Driven Approach to Cleaning Large Face Datasets". [Online] Available: http://vintage.winklerbros.net/Publications/icip2014a.pdf. [Accessed: 23 September 2016].

[20] G. E. Hinton, R. R. Salakhutdinov. "Reducing the Dimensionality of Data with Neural Networks". *In Science* vol 313, pp. 504-507. July 2006. DOI: 10.1126/science.1127647

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens. "Rethinking the Inception Architecture for Computer Vision". arXiv:1512.00567v3 [cs.CV] 11 Dec 2015

[22] C. Szegedy, S. Ioffe, V. Vanhoucke. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning". arXiv:1602.07261v1 [cs.CV] 23 Feb 2016

[23] Y. Jia, E. Shelhamer, J. Donahue, et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". [Online] Available: http://ucb-icsi-vision-group.github.io/caffe-paper/caffe.pdf. [Accessed: 27 September 2016]

[24] Z. Liu, P. Lui, X. Wang, X. Tang. “Deep Learning Face Attributes in the Wild”. arXiv:1411.7766v3 [cs.CV] 24 Sep 2015

[25] X. Zhang, L. Zhange, x. Wang, H. Shum. “Finding Celebrities in Billions of Webpages”, to appear on IEEE Transaction on Multimedia, 2012 [Online] Available: http://research.microsoft.com/en-us/um/people/leizhang/paper/TMM2011_Xiao.pdf. [Accessed: 27 September 2016]

Glossary

AJB: Advanced Joint Bayesian

ANN: Artificial Neural Network

BGR: Blue Green Red

CNN: Convolutional Neural Network

CPU: Central Processing Unit

CV: Computer Vision

DA: Data Augmentation

EM: Expectation Maximization

GB: Gigabyte

GPI: Image Processing Group

GPU: Graphics Processor Unit

ILSVRC: ImageNet Large Scale Visual Recognition Competition

JB: Joint Bayesian

LFW: Labeled Faces in the Wild

ML: Machine Learning

MSE: Mean Squared Error

MSLE: Mean Squared Logarithmic Error

PCA: Principal Component Analysis

RAM: Random Access Memory

RBF: Radial Basis Function

ReLU: Rectified Linear Unit

RGB: Red Green Blue

ROC: Receiver Operating Characteristic

SGD: Stochastic Gradient Descent

SVM: Support Vector Machine

TF: TensorFlow

TSC: Department of Signal Theory and Communications

TV: Television

UPC: Universitat Politècnica de Catalunya

VGG: Visual Geometry Group

WP: Work Package

YTF: YouTube Faces

ZCA: Zero-phase Component Analysis