Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Incremental Class Representation Learning for Face Recognition

Degree's Thesis
Audiovisual Systems Engineering

Author: Eric Presas Valga
Advisors: Elisa Sayrol, Josep Ramon Morros

Universitat Politècnica de Catalunya (UPC)
2016 - 2017

Abstract

Image classification is one of the most active challenging problems in computer vision field. Taking this to Deep Neural Networks with systems that are able to deal with large data sets as it can be ImageNet. Large Convolutional Networks as VGG-16 used in this work have recently demonstrated impressive classification performances.

This work is focused on novel techniques for Incremental Learning stages for face recognition, which is an important open problem in artificial intelligence. The main challenge of this work is the development of incrementally learning systems that learn about more and more concepts over time.

Most of the actual methods that use incremental learning in *"online"* or *"offline"* stages. This thesis focuses on *"offline"* incremental stages where the data available is distributed in batches of classes. Since the necessity to deal with a continuous training stages, some well-established methods for transfer learning are applied by the author to run the experiments.

Preserving knowledge is the most challenge task to deal using incremental learning techniques. The actual research is on apply incremental learning in natural systems where for example, it is not considered to store all the old training data to make a new model when new data comes available.

Another interesting concept for incremental learning systems is *"lifelong"* learning, which are related to the methods analyzed in this work since the system proposed also learn from a sequence of different tasks. The similarity of multi-task learning and *"lifelong"* learning is that they both use shared information across tasks to help learning, but also, multi-task learning is not able to grow the number of tasks over time preserving the knowledge.

Resum

La classificació d'imatges és una de les tasques més desafiants en el camp de la visió per a computadors. Portant això al camp de les xarxes neuronals profundes utilitzant sistemes que són capaços de gestionar datasets considerablement grans, com pot ser el de ImageNet. Grans xarxes convolucionals com pot ser VGG-16, que és la que s'utilitzarà en aquest treball i que ha demostrat molt bons resultats.

Aquest treball està focalitzat en noves tècniques per aprenentatge incremental per al reconeixament de cares, que és un important problema obert en la inteligència artificial. El major repte en aquest treball és desenvolupar dos sistemes incrementals que aprenen més conceptes durant el temps.

Molts dels actuals mètodes que utilitzen l'aprenentatje incrmental en escenaris com*"online"* o *"offline"*. Aquest treball està focalitzat sobretot en els sistemes incrementals que utilitzen *"offline"* com a mètode incremental d'aprenentatge on les dades són proporcionades per conjunts de classes, on cada conjunt apareix en un moment diferent. Hi ha una necessitat per a gestionar amb escenaris d'aprenentatge continuu, i és per això que mètodes de tranferència d'aprenentatje sert'an estudiats i implementats per l'autor del projecte per tal d'executar els experiments.

Una de les tasques més desafiants és com gestionar i preservar el coneixament obtingut per tal de no oblidar. Quan es parla de aprenentatje incremental, molts cops està relacionat amb el concepte de sistemes naturals on per exemple, no està contemplada la possibilitat de guardar totes les mostres del coneixament adquirit per a un futur entrenament quan hi hagin noves classes disponibles. D'altra banda, l'aprenentatge *"online"* es diferencia del *"offline"* durant el procés d'entrenament. On s'encarrega d'apendre de forma eficient amb dades que arriben de forma incremental però sempre per les mateixes tasques, dit d'altre forma, els sistemes que utilitzen l'aprenentatge *"online"* en la majoria de treballs proposats, no s'encarreguen d'incrementar el nombre de classes.

Un altre concepte interessant per als sistemes d'aprenentatge incremental és el que se'n diu aprenentatge *"lifelong"*, que també està relacionat amb els mètodes analitzats en aquest treball, ja que el sistema proposat també aprèn d'una seqüència de tasques diferents. També hi ha una similaritat entre l'aprenentatge per múltiples tasques i l'aprenentatge *"lifelong"*, que és que els dos utilitzen informació compartida entre tasques per ajudar en l'aprenentatge, de totes formes, els sistemes d'aprenentatge per a múltiples tasques tampoc poden augmentar el nombre de classes.

Resumen

La clasificación de imágenes es una de las tareas más desafiantes en el campo de la visión por computador. Llevando esto al campo de las redes neuronales profundas utilizando sistemas que són capaces de gestionar datasets considerablemente grandes como puede ser ImageNet. Grandes redes convolucionales cómo puede ser VGG-16, que és la que se utilizará en este trabajo, han demostrado muy buenos resultados.

Este trabajo está focalizado en nuevas ténicas para aprendizaje incremental para el reconocimiento de caras, que és un importante problema abierto en la inteligencia artificial. El mayor reto en este trabajo consiste en desenvolupar dos sistemas incrementales que aprenden más conceptos a medida que pasa el tiempo.

Muchos de estos métodos que utilizan el aprendizaje incremental en escenarios cómo *"online"* o *"offline"*. Este trabajo está focalizado sobretodo en los sistemas incrementales que utilizan *"offline"* como método incremental de aprendizaje dónde los datos son propocionados por conjuntos separados de classes. Hay una necesidad clara de gestionar escenarios de aprendizaje continuo, y es por este motivo que métodos de transferéncia de aprendizaje han estado estudiados y implementados por el autor del proyecto para tal de llevar a cabo la ejecución de experimentos.

Una de las tascas más desafiantes es cómo gestionar y preservar el conocimiento obtenido para no olvidar. Cuando se habla de aprendizaje incremental, muchas veces va relacionado con el concepto de sistemas naturales dónde por ejemplo, no está contemplada la opción todas las muestras para el conocimiento adquirido para un futuro entrenamiento cuando haya clases disponibles para hacerlo. En cambio, el aprendizaje *"online"*, se diferencia del *"offline"* durante el proceso de entrenamiento. Dónde se encarga de aprender de forma eficiente con datos que llegan de una forma incremental peró siempre corresponden a las mismas clases, dicho de otro modo, los sistemas que utilizan el aprendizaje *"online"* en la mayoria de trabajos propuestos, no se encargan de incrementar el nombre de clases.

Otro concepto interesante para los sistemas de aprendizaje incremental es lo que se llama aprendizaje *"lifelong"*, que también está relacionado con los métodos analizados en este trabajo, ya que el sistema propuesto también aprende de una sequéncia de tascas distintas. También hay una similitud entre el aprendizaje para múltiples tascas i el aprendizaje *"lifelong"*, que es que los dos métodos utilizan información compartida entre tascas para ayudar en el aprendizaje, de todas formas, los sistemas de aprendizaje para múltiples tascas tampoco puede augmentar el nombre de clases.

Acknowledgements

First of all, I would like to express my gratitude to my supervisors, Elisa Sayrol and Josep Ramon Morros, for giving me the opportunity to work on this trending topic. Also for their patience guidance during the project.

I also wish to thank to the Image Processing Group (GPI), specially to Albert Gil and Josep Pujal for giving me acces to the Development Platform of the UPC.

Revision history and approval record

Revision	Date	Purpose
0	20/06/2017	Document creation
1	29/06/2017	Document revision
2	30/06/2017	Document revision
3	30/06/2017	Doccument approbation

Name	E-mail
Eric Presas Valga	eric.presas@alu-etsetb.upc.edu
Elisa Sayrol	elisa.sayrol@upc.edu
Josep Ramon Morros	ramon.morros@upc.edu

Written by:		**Reviewed and approved by:**		**Reviewed and approved by:**	
Date	20/06/2017	**Date**	30/06/2017	**Date**	30/06/2017
Name	Eric Presas Valga	**Name**	Elisa Sayrol	**Name**	Josep Ramón Morros
Position	Project Author	**Position**	Project Supervisor	**Position**	Project Supervisor

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Computer Vision is trying to emulate the human visual system to extract useful information from images or video sequences. Image classification is one of the most essential tasks in computer vision. Since Perceptron Neural Networks appears in early 70's, the challenge was to train networks with multiple Layers for Computer Vision classification tasks. LeCun applied the back propagation algorithm [26] to a deep neural network for recognizing and classifying handwritten digits[13].

Over the last two decades lots of advances in the Computer Vision field can be appreciated. The availability of ImageNet results [5] show that Deep Neural Networks achieve a great success on recognition tasks. Works using Deep Neural Networks for face recognition purposes have been published over last decade. These systems are achieving also a great success (i.e DeepFace network).

The motivation of Incremental Learning is to make learning systems able grow the number of tasks learned in time continuously adding new data for more tasks without forgetting the knowledge obtained.

Incremental classifiers using Deep Neural Networks for adding new capabilities to a system in computer vision stages are under research since the last few years. These kind of classifiers have to preserve the knowledge for the old tasks while acquiring knowledge for the new tasks at the same time.

In this final degree project, I'm going to use a well-established Deep Convolutional Neural Network model, as it is VGG [22] and apply some methods [20, 14] to make the system able to learn tasks at different moments using streams of data. This work is focused on Face Recognition and how to use this state-of-the-art methods which are well explained in Sections 3, 2.5.

1.1 Motivation

Lots of visual applications actually using Deep Convolutional Neural Networks to perform some classification tasks have been proposed in recent years. When building a gradually growing system, new capabilities are continuously added. There are two kind of methods, the ones that store representations from the classes that have been learned and the other ones, that do not need to store any sample from the original tasks.

Training large-scale data sets from scratch needs a lot of computational time and a lot of memory available, that's why it's interesting to research on incremental learning methods. This project aims to use incremental learning methods for recognizing faces. The entire training process is separated into several parts. Each part will train the system with its correspondent classes.

This project's motivation is about to deal with data sets that are streams of data, new visual information will be continuous incorporated while existing knowledge will be preserved. Taking these ideas [14, 20] we build a fage recognition system.

1.2 Objectives

The first purpose of this project consists on analyzing how Deep Convolutional Neural Networks work in stages such as face recognition, which is heavily exploited to build identity verification and security surveillance systems. This research propose is to build a system based on the related works 3. The system will be able to increment its number of classes giving a competitive multi-task classifier after every training.

The content of this thesis, will be based on the related works that have been published so far, and that's why the author is going to investigate through the proposed methods. These methods will be implemented and tested in order to extract some conclusions about its performance on face recognition. Also, topics like learning visual features, incremental learning and knowledge distillation using Deep Convolutional Neural Networks will be covered in this project. All of the concepts explained in the related papers will be tested and evaluated by the author to design a competitive model for face recognition on an incremental learning stage, in other words, the system has to be able to recognize faces over a stream of data with a certain accuracy.

Also, an analysis for selecting parameters and apply methods to outperform the incremental class representation is provided, and furthermore the evaluation for the limitations of this methods and a research of all the related methods that have been published but don't perform the actual state-of-the-art.

1.3 Requirements and Specifications

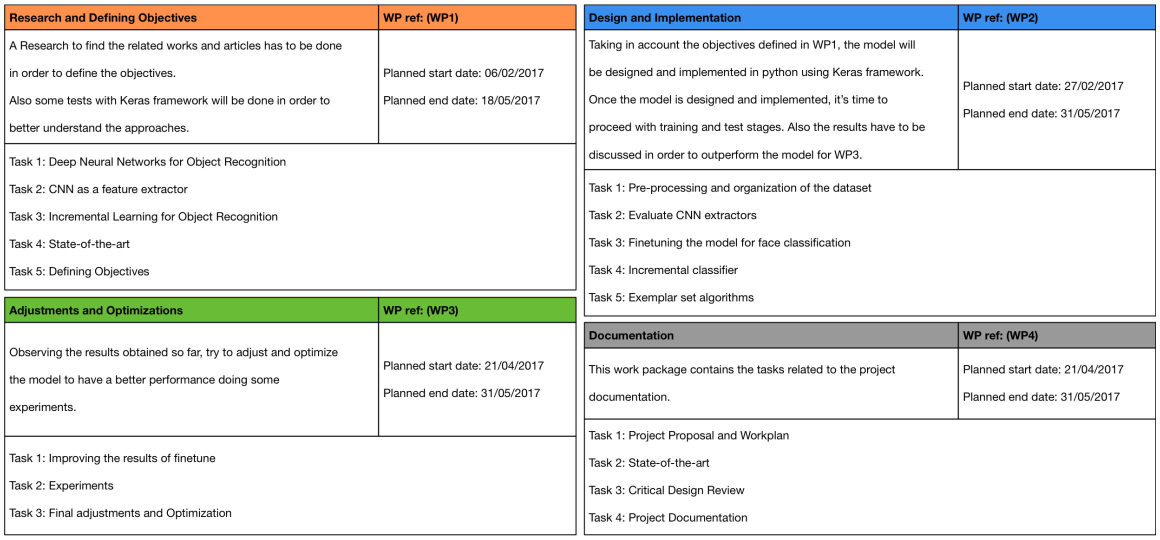
To implement the algorithms and reproduce the experiments that will be done over the methods that are going to be explained in next sections, a workspace with the required software has to be chosen and installed. In this thesis, all the experiments require of Python 3.4 and Keras 1.2.2 over Tensorflow GPU version 1.0.0 libraries followed by NumPy to deal with data vectors and matplotlib libraries to plot the results. Also, since a GPU will be used, some extra modules are needed: Cuda 8.0 with cuDNN v5.1 to accelerate the processes and the GPU.

Hardware requirements to work with Keras and Tensorflow are carried out at the Image Processing Group at UPC lab, providing to the author an access to the Development Platform where the required RAM memory and a GPU memory is available.

The author has to two systems with the requirements seen before, using these methods [14, 20] to make a study of its performance on face recognition. As a specification, a detailed analysis will be done, including some experiments to show the performance of Incremental Learning using VGG network on face recognition.

1.4 Work Plan

In Figure 1 a general overview of the work packages and its internal tasks are explained, also, the Gantt Diagram related to the time expended on each task is included in Figure 2.



Research and Defining Objectives	WP ref: (WP1)
A Research to find the related works and articles has to be done in order to define the objectives. Also some tests with Keras framework will be done in order to better understand the approaches.	Planned start date: 06/02/2017 Planned end date: 18/05/2017
Task 1: Deep Neural Networks for Object Recognition Task 2: CNN as a feature extractor Task 3: Incremental Learning for Object Recognition Task 4: State-of-the-art Task 5: Defining Objectives	

Design and Implementation	WP ref: (WP2)
Taking in account the objectives defined in WP1, the model will be designed and implemented in python using Keras framework. Once the model is designed and implemented, it's time to proceed with training and test stages. Also the results have to be discussed in order to outperform the model for WP3.	Planned start date: 27/02/2017 Planned end date: 31/05/2017
Task 1: Pre-processing and organization of the dataset Task 2: Evaluate CNN extractors Task 3: Finetuning the model for face classification Task 4: Incremental classifier Task 5: Exemplar set algorithms	

Adjustments and Optimizations	WP ref: (WP3)
Observing the results obtained so far, try to adjust and optimize the model to have a better performance doing some experiments.	Planned start date: 21/04/2017 Planned end date: 31/05/2017
Task 1: Improving the results of finetune Task 2: Experiments Task 3: Final adjustments and Optimization	

Documentation	WP ref: (WP4)
This work package contains the tasks related to the project documentation.	Planned start date: 21/04/2017 Planned end date: 31/05/2017
Task 1: Project Proposal and Workplan Task 2: State-of-the-art Task 3: Critical Design Review Task 4: Project Documentation	

Figure 1: Work Packages and internal Tasks

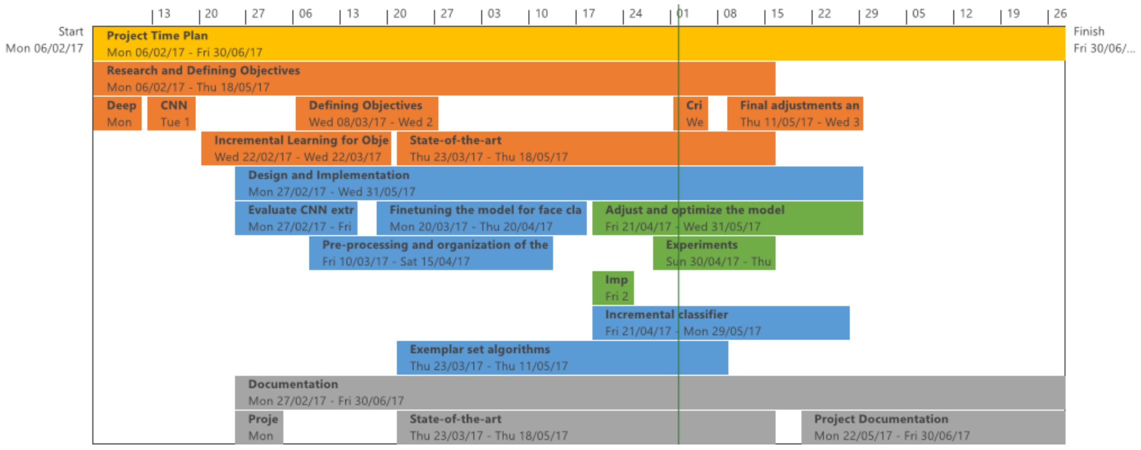


Figure 2: Gantt diagram

1.5 Deviations

Learning how to work with the required software was a heavy part of the project since the author didn't know about the usage of the libraries before the project started. So this part was extended a couple of weeks.

Otherwise, since there aren't any publications of the codes related to the papers, the author had to deal with some coding problems, what also delays the implementation stage. Also, the firstly chosen data set which was Labeled Faces in The Wild (LFW) was not working properly. We conclude that the LFW data set won't work properly in incremental face recognition learning stages, because its lack of images in some classes, that in a lot of cases are represented by a single image. An incremental learning system can't learn from a class with a single sample and that's why the change to FaceScrub data set.

1.6 Document Structure

The rest of the content of this thesis is structured as following. Section 2 describes literature reviews on image classification focusing on deep structures using Convolutional Neural Networks to later introduce the concept of Incremental Learning which is explained in details in Section 2.5. Section 3 presents the related work that perform the actual state-of-the-art as described in the respective papers. Section 4 shows the detail of the experiments done in face recognition with two methods and some modifications made by the author to improve the results. An analysis of the Budget for this research project is shown in 5. Finally the author's conclusions and a future work are included in Section 6.

2 Literature review

Deep Learning emerged from the sub field of artificial intelligence that uses neural networks to perform some tasks. Deep Neural Networks are feedforward networks with many hidden layers. The number of hidden layers to consider the network as a *"deep"* structure is not defined. Instead, networks with one hidden layer are considered as *"shallow"* structures.

Interesting applications using deep learning can be found in fields like computer vision, natural language processing and also speech/audio processing.

A basic explanation of the baseline for the methods used on face recognition stages is done in Section 2.1 to 2.4, to later focus in Section 2.5 on the main idea of incremental learning and its variances comparing to the novel methods presented in this work, Section 3.

2.1 Deep Neural Networks

To explain how deep neural networks work, some concepts will be reviewed taking in account that the reader is familiarized with Deep Neural Networks. A brief introduction to Neural Networks followed by its applications in deep architectures can be found in this section. Also, Convolutional Neural Networks will be explained specifically for object/face recognition tasks.

Neural Networks

A neural network is an interconnected group nodes or neurons where each of group nodes are called layers. Each neuron is a computationally unit that takes several inputs $x_1, x_2, \ldots, x_N$, with its correspondent outputs *f(x)*, seen on left-side of Figure 3 , which is called the activation function that can be for example *rectifier linear unit* (mostly used in this work), *tanh*, *sigmoid* and more. In Figure 3 an overview of the simplest neural network with one hidden layer is shown, as it's said, every neuron has one connection to all the neurons of the next layer. This kind of networks are called fully-connected.

The neurons are connected by links and interact with each other. Using the back-propagation algorithm explained in [26], the network learns by example. That means if an example of what the network has to do is submitted to the algorithm, it changes its weights so the desired output for a particular input can be produced.

The back propagation algorithm is one of the most popular Neural Network algorithms that can be separated to four main steps, which are Feed-forward computation, back propagation to the output layer, back propagation to the hidden layer and network weight updates as described in [3].

Feed-forward pass is a process composed by two steps. First part is taking the values of the hidden layer nodes, while the second part is to use those values from hidden layer to compute values of the output layer. Hidden layer's values are multiplied with weights of connecting nodes. Once the values from hidden layers are multiplied by the weights of nodes, next step is to calculate the error obtained in the output layer. When error is known, it will be used for backward propagation to adjust the weights.

The error has to be propagated from hidden layer down to the input layer to later update the weights. Now having calculated the output layer forward pass, the error has to be computed.

On classification stages, in a speech/audio or image/video recognition problem, usually needs kind of feature extractor function $\phi()$. This kind of networks can't extract good features from a complex amount of raw data.

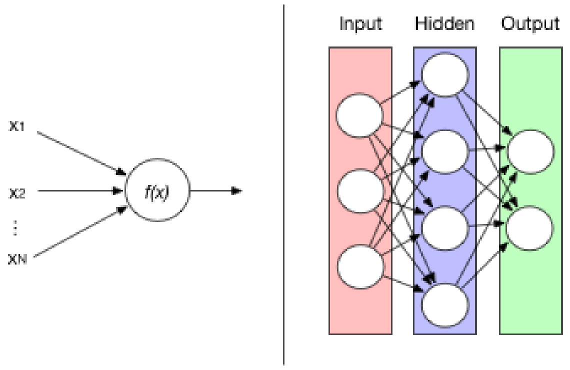


Figure 3: A perceptron and a simple neural network structure

Deep Neural Networks

Deep Neural Networks are artificial neural networks that contains more than one hidden layer as Figure 4 shows. The main idea is to keep away the process of designing hand-crafted features. Most of this kind of networks deals with raw data.

To simulate the process of feature extraction, these networks use a cascade classifier of layers of nonlinear processing units. It is useful to learn multiple levels of representations, so the levels form a hierarchy of concepts.

It's quite interesting this part of learning the data representations, for example, an image can be represented in many ways. Some representations are better than others at simplifying specific tasks (i.e face recognition). Deep Learning solves that problem not by generalizing the feature extractor to do a certain task which is basically how typical hand-crafted features methods work, instead Deep Neural Networks are able to learn how to extract features.

Deep Learning architectures such as deep convolutional neural networks [22, 19, 25], deep belief networks [9], or recurrent neural networks [24] have been applied to fields like computer vision, automatic speech recognition, natural language processing and bioinformatics where the produced results are in some cases superior to human experts.

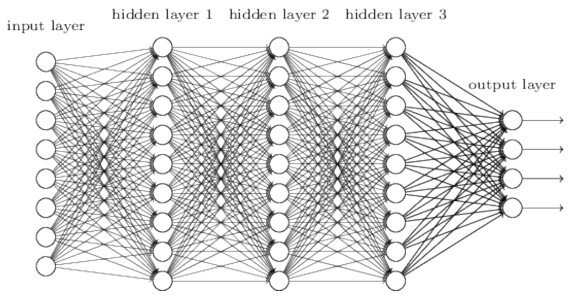


Figure 4: A deep neural network structure

2.2 Convolutional Neural Networks

Fully connected layers from neural networks do not take into account the spatial structure of the data, which provide a lot of information about the image. The spatial structure of the image has to be preserved, to make it possible, different convolutions are done all over the image. To provide not also the spatial information but some additional information (i.e shapes, color, etc). Convolutional Neural Networks combine three hierarchical ideas to ensure some degree of shift, scale and distortion invariance. The network is trained like standard neural networks by back propagation algorithm. It was firstly proposed by Yann LeCun, LeNet [13] that used neural networks with convolutional layers followed by a fully-connected block of layers to classify as can be seen in Figure 5.

After LeNet was published, there have been several deeper architectures published in recent years that improves the performance, but still using as a basis, the main concepts from LeCun's work. Convolutional Neural Networks alternate between convolution layers with non-overlapping sub-sampling layers. The convolutional layers are used to extract features from local receptive fields. These layers are organized in planes of neurons called feature maps, which are responsible to detect a specific feature. In Section 2.3.3, some heavily exploited deep convolutional neural networks for face recognition tasks are introduced.

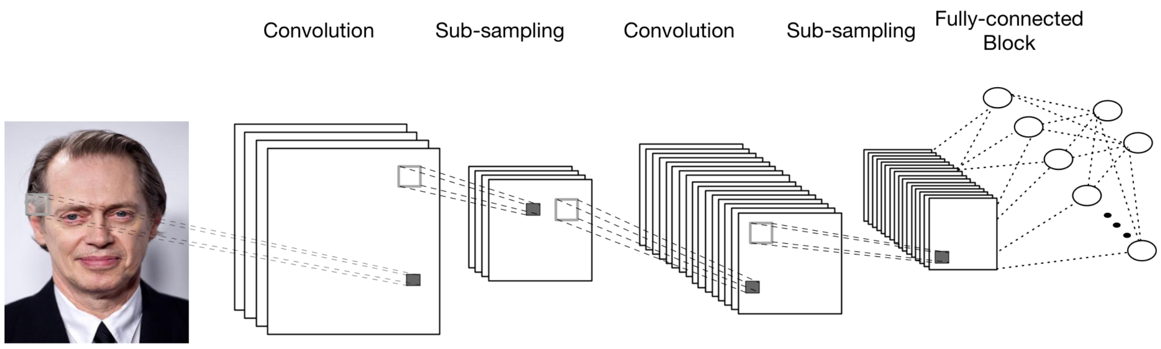


Figure 5: A Simple Convolutional Neural Network Structure

2.3 Face Recognition

Recognizing Faces from images or videos is a popular trending topic in the computer vision research field. Many places and buildings have surveillance cameras for security proposes. Face recognition systems are playing an important role in this field. Also, it is typically used in access control systems by identity analysis which can be compared to biometrics systems like fingerprint or eye iris recognition systems.

Face recognition is a topic that is receiving a significant attention with lot of approaches proposed. All these methods can be distinguished in two main groups. The ones that do not use deep structures "shallow" which ones that do "deep" as referred in [19].

Shallow methods are introduced in next subsections taking into account that the explanation will be focused on deep architectures, what is the exploited theme in this work. Most face Recognition systems have three main steps: pre-processing, feature extraction and classification as shown in Figure 6.

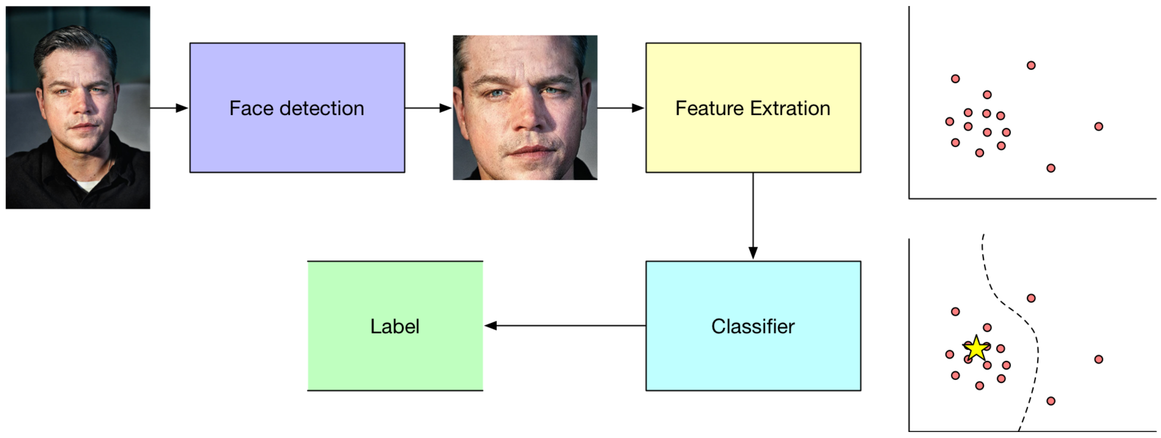


Figure 6: Diagram of a simple face recognition system

2.3.1 Pre-processing

While building a Face recognition system a good data set neds to be chosen to train the system achieving a good performance. Also, when implementing these systems in a real situation where there are for example in access control cameras, most of the times the image to be analyzed is not just the face or the lighting conditions are not always the same. In these cases some preprocessing algorithms are needed.

In deep architectures we don't really care about the lighting conditions, instead it is preferred to have the cropped version of the image containing just the face in our case. Usually a face recognition systems take use of face detection algorithms like Haar cascade classifier proposed by Viola-Jones in [27] or this approach using Convolutional Neural Networks proposed by the Yahoo team in [6]. So using face detection before the recognition stage make the system able to recognize multiple faces from an image. Face detection methods sometimes are a preprocessing for face recognition algorithms.

2.3.2 Feature Extraction

As it has been said, we can differentiate between shallow and deep structures. There is a basic difference between this two kind of methods, that is the step of feature extraction. For shallow structures, there are some well established algorithms for example Eigenfaces [23], that are based on the dimensionality reduction approach of Principal Component Analysis (PCA). Fisherfaces [16], which approach is based on Fisher's famous Linear Discriminant Analysis (LDA). Also some approaches like SIFT [15] or SURF [1] used before for image matching, were applied in some works for this part of feature extraction in recognition systems.

Deep structures do not need hand-crafted features, it means that using a Convolutional Neural Network as a learning feature extractor can solve the problem, seen in Section 2.2. Actually Convolutional Neural Networks architectures are achieving the actual state-of-the-art accuracy for face recognition. DeepFace from Facebook research group proposed in [25] or VGGFace proposed in [19] from Oxford University.

2.3.3 Classification

Once features from all samples are extracted, the next step is to assign a class to the image. Sometimes in typical face recognition systems two or more classifiers are combined from a wide variety of classification methods when using a hand-crafted feature extractor. The mainly exploited classifiers for supervised learning, can be *Support Vector Machines*.

In deep architectures, the classification stage is more or less the same for every related work. The part of feature extraction can be differentiated by two commonly used methods as Figure 7 shows.

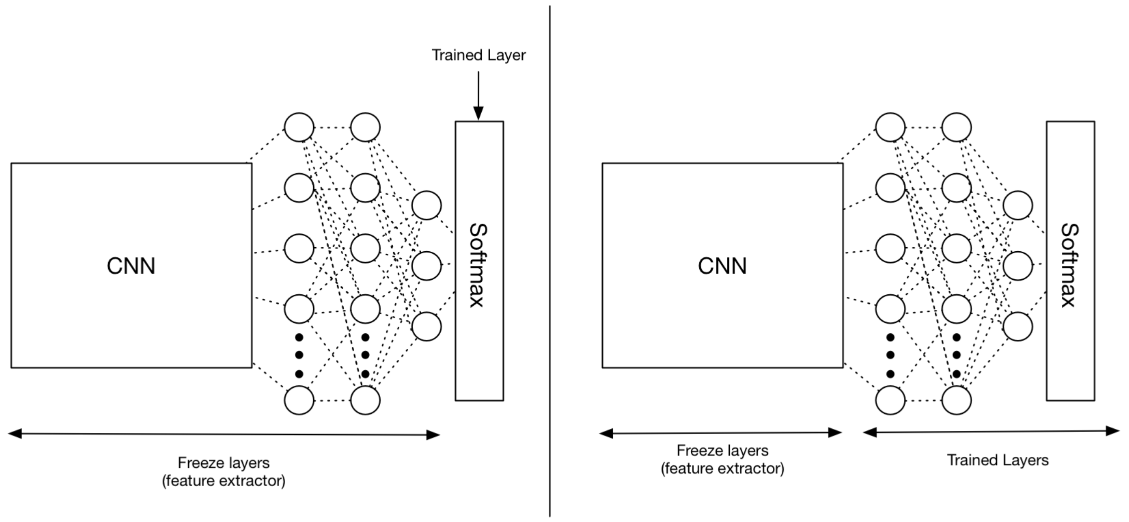


Figure 7: Two different methods for defining the feature extractor

There are several deep neural network architectures performing the actual state-of-the-art in face recognition. Most of the times, the classification part is a fully-connected block containing two or three dense hidden layers followed by a softmax classification layer. Some examples are DeepFace from Facebook research group [25] and VGGFace, which is the commonly used VGG-16 pre-trained for faces by Oxford University [19]. VGG possible architectures are shown in Figure 8, note that VGG-16 corresponds to the *D structure*, which is the network structure used to run the experiments in this work as Section 4 establishes.

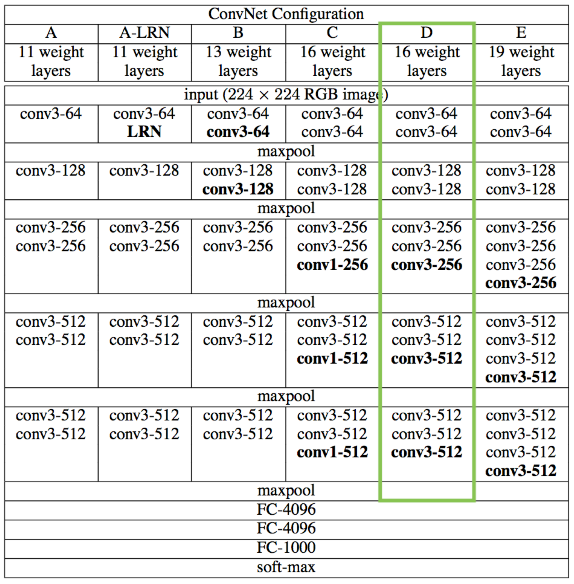
2.4 Understanding Convolutional Neural Networks

In this Section, we introduce a brief explanation of what convolutional layers from CNNs learn in convolutional blocks. As described in Section 2.2, each convolution layer corresponds to a level of features.

Several approaches for understanding and visualizing Convolutional Neural Networks have been developed recently [30] in order to interpret what convolutional layers are learning. The most straight-forward visualization technique is to examine the activations of the network during the forward pass.

These methods to analyze the network are useful to detect anomalies over the training process. For example, if a high learning rate is chosen in training, the visualizations on some activation maps may be zero for many different inputs, which can indicate what is called *dead filters*. In Figure 9, three randomly chosen filters weight outputs for each convolutional block are shown for four identities.

Convolutional Neural Networks use learned filters to convolve the feature maps from the previous layer. To invert this procedure, a deconvolutional network is used, which is basically the inverse of the convolutional network using the transposed versions of its filters applied to the rectifier maps, in other words, the filter have to be flipped vertically and horizontally as describe it in [30].



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					

Figure 8: Table of VGG architectures, in this work the architecture "D is selected"

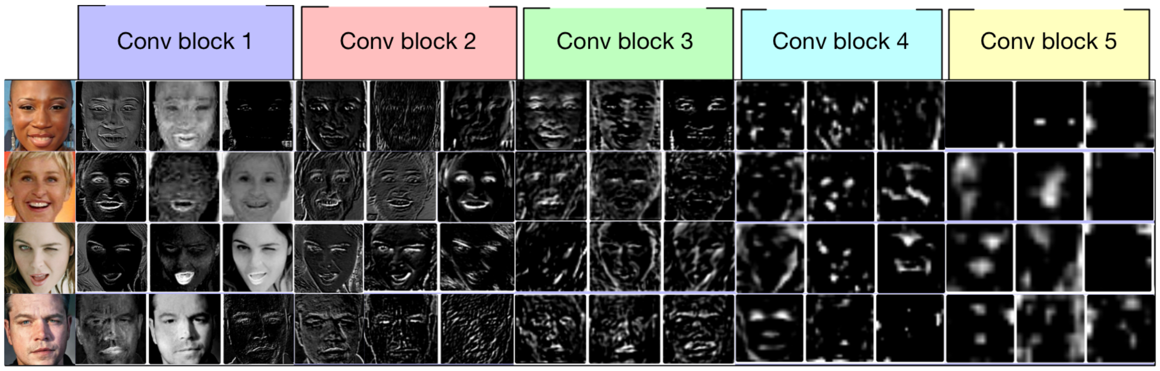


Figure 9: Randomly chosen weight layer outputs

2.5 Class Incremental Learning

Incremental Learning considers systems that can learn new tasks over a trained system, the main goal is to retrain sequentially the model with new knowledge without losing the knowledge achieved before for the old tasks.

Some concepts about transfer knowledge between networks are introduced in this Section and also an explanation of how incremental learning works and its different proposed approaches in recent years.

Most of the methods proposed for continuous integration of new tasks to a system has to accomplish the following. The resulting model has to preserve the old tasks performance while learning for new ones. Some additional concepts that the author is going to study like Distillation Knowledge and incremental fine-tuning are used to preserve the knowledge.

2.5.1 Fine-tuning

Transfer learning and fine-tuning methods are heavily exploited and always for the same reason. Usually the data set size is not sufficient for the depth of the network. Then, these methods require of a pre-trained model as a initialization. Usually the top layers are removed and replaced to adapt the network to the presented problem.

It's commonly used to pre-train a Deep Convolutional Neural Network with a very large data set (i.e. ImageNet), and then use the weights of this model as a initialization of the new network as said in [11]. There are many ways for transfer learning, these are presented below.

Convolutional Neural Network as a feature extractor

Once a pre-trained Deep Convolutional Neural Network is available the top layer is replaced by a new one having the desired outputs to solve the problem. In training stage, just the top layer recently added has to be trainable and taking the rest of the network frozen as left-side of Figure 7 shows. Training the rest of the network is computationally heavy and don't give better results at all.

We can differentiate the feature extractor in two groups, the ones that contains the fully connected part of the network, and the ones that not include any fully connected layer. The last ones takea just the feature maps of the last convolutional layer which are called *"bottleneck features"*, seen also in Figure 7 on the right-side.

Fine-tuning the Convolutional Neural Network

This strategy is the same as explained in last point, but it's not only about replacing the top layer and retraining it freezing the networks for the other layers. Here in this case, all the network is retrained, also instead of training all the layers, the layers that are desirable to be trained can be setted up as trainable.

Mostly the first convolutional layers are frozen, because as observed in Section 2.4, the earlier features of the CNN are more generic (i.e edge detectors, color detectors...) and that's why the training is fixed in later layers, that content more specific details of the classes contained in the original data set. Figure 10 shows an example of the commonly used way to finetune the VGG-16 network and what is done in this work. Typically, the last convolutional block is set up as trainable like all the fully connected block, so the network is learning how to extract the last features for the chosen data set.

Pre-trained models

Since the actual Deep Convolutional Neural Networks takes 2-3 weeks to train across multiple GPUs on ImageNet data set, there are a lot of shared models around the web. It's a common practice to take a model trained by someone that does similar tasks and fine-tune it with the methods explained to solve our problem. In Figure 7 a schema for using the Convolutional Neural Network as a feature extractor is provided followed by the proposed architecture for fine-tuning the network used in this thesis as Figure 10 shows.

2.5.2 Distillation Knowledge

Distillation knowledge [8] is commonly used to squeeze the knowledge of a trained network into a smaller one by using the outputs of the trained network as a soft targets instead of using the logits for every class, seen in $q_i = \frac{exp(g_i)}{\sum_{j}^{n} exp(g_i)}$ where it is reflected to the loss function when training $l(\theta) = -\sum_{i=1}^{n} \delta'_i * log(q_i)$, as

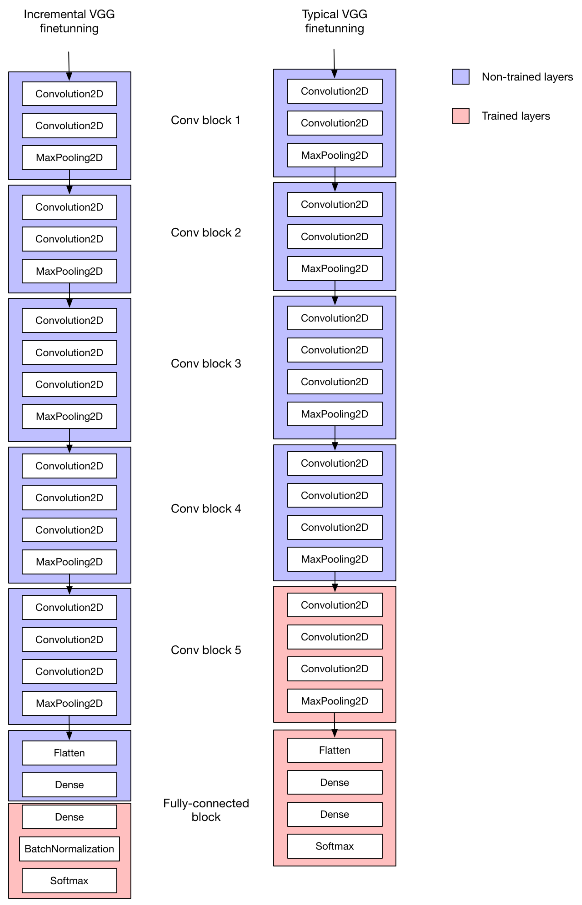


Figure 10: fine-tuning layers for this work is shown in left-side and the typical fine-tuning stage for VGG is shown on right-side

described in [referencia distillation] where the logits are changed by $\delta_i' = \frac{(\delta_i)^{\frac{1}{T}}}{\sum_j (\delta_j)^{\frac{1}{T}}}$

In the simplest form, this is also a case of transfer learning, the knowledge is transferred from one network to another. Like every transfer learning stage, the knowledge of the model is preserved. When training the new model, a transfer set is used, which consists on using a soft target distribution for each case, so the new network is feeded by the old network outputs, that's why the loss function do not look to the logits anymore . Using this method the knowledge will be preserved as we can see [8].

2.5.3 Class Batch Learning

since the necessity for dealing with continuous data streams is vast, there is a growing number of approaches which replace fixed models with ones that can be able to adapt themselves continuously to the stream data set as Figure 11 shows.

A frequently used approach called online learning can be seen as a special case of continuous learning, which data set is not presented at once as seen in [2]. This approaches take data from online sources and grow up the number of classes itself if some strong data to learn from is available, usually using Support Vector Machines as a model to classify.

Taking this models to Deep Neural Networks field, lots of approaches are studied and are constantly

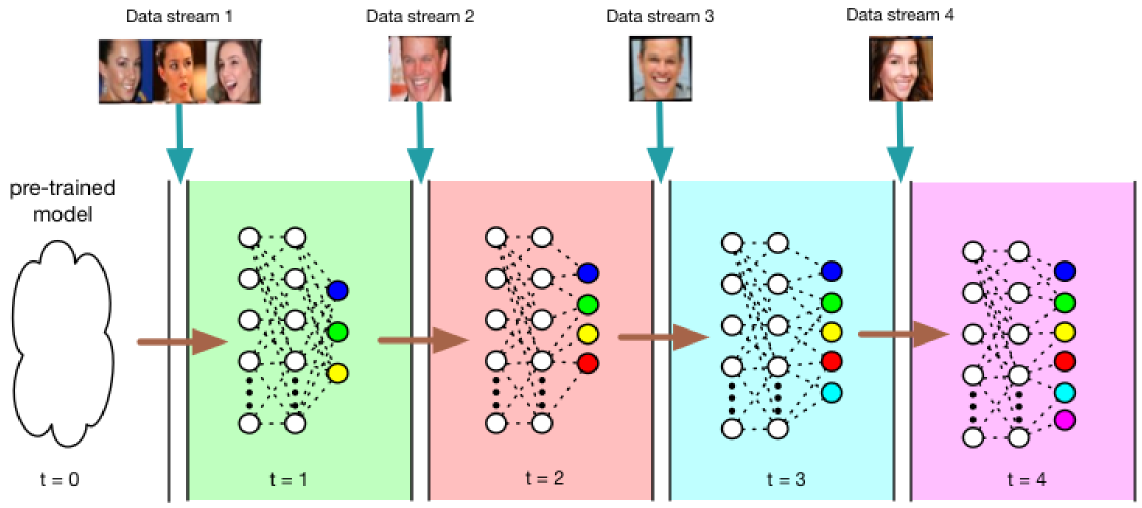


Figure 11: Model that adapts itself to new classes

under research. A decade ago, a published work done by Wilson and Martinez that compares batch learning and online learning in neural networks [28]. They said that there are no practical advantage of learning with mini-batches over online learning.

Novel proposed approaches like [29] deal with stream data sets duplicating the existing network, then fine-tuning each network individually to later decide which of two networks has the better performance. This method is relatively complex, memory-intensive and computationally demanding.

In this work, some novel approaches of incremental class learning will be introduced. These approaches do not use the previous classes information to finetune the network for new tasks as seen [14, 20, 21, 10]. Some topics like Incremental fine-tuning, catastrophic forgetting and distillation are the baseline of this novel approaches [20, 14]

2.5.4 Incremental fine-tuning

When talking of fine-tuning, we are talking about a kind of incremental learning where there is an initial pre-trained model θ_t. Then for training to the new tasks θ_{t+1}, the initial model θ_t is taken in order to have the weights of the network well initialized.

In fine-tuning stages, it's required that the parameters do not vary too much keeping the learning rate as low as possible to avoid rare performances on training stage.

Incremental learning uses fine-tuning in a different way, instead of replacing the top layer to adapt the model to the new tasks, more nodes are added to the top layer to increase the number of classes to be predicted. In other words, a new layer correspondent to the number of new classes is concatenated with the old one, that contains the classification information for the old tasks.

The knowledge for the old classes has to be preserved while adding knowledge for the new ones. Some approaches to deal with catastrophic forgetting in [12, 7], which basically take some information about the old network outputs or using dual-network models [29].

2.5.5 State-of-the-art Incremental Learning

Multi-Task learning has been a field of research in the last few years, this systems are able to learn different tasks jointly by exploiting its commons and differences. The tasks are learned in parallel while using shared representations, meaning that the knowledge learned for a task can help other tasks to improve its performance.

Lifelong Machine Learning Systems, which basically differentiates from Multi-Task learning by it's incremental and continuous learning property. Lots of related works in this field using the transfer knowledge to learn for new tasks [8, 17, 11, 10].

Incremental learning systems can be categorized in two general approaches that trains deep networks from data streams. The first approach is online learning, here the data set used is not static, so it's typical to have more data but also time constraints. Usually this method is also called online fine-tuning because the learning of the network is based on Sigmoid Gradient Descendent (SGD). The deep network is continuously fine-tuned with new data as the data is accumulated [17].

The second approach is called offline learning or batch learning, where the algorithm updates the network parameters after consuming the whole batch of classes, instead of online learning where the algorithm update the network parameters after learning from one training instance.

To deal with incremental learning stages there are basically two main used methods, which are dual-

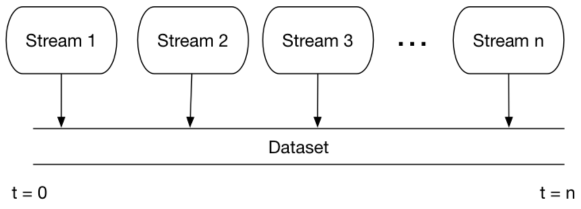


Figure 12: Example of a stream of data that comes over time

network memory models and incremental fine-tuning models. Using a dual-network to deal with the learning for new classes was heavily exploited in the last 90's, and which performance is improved by Motonobu Hattori in 2009 [7] heavily inspired by the complementary learning systems theory.There are two networks, one with a small storage capacity and another one with a recurrent structure.

The second method is incremental fine-tuning, which is related to this work. Two interesting studies that use this method are presented below [20, 14]. These systems basically are taking into account the knowledge learned for the classes.

3 Related Work

In this Section is explained what the author have implemented, and why this methods [14, 20] are chosen. The author will take use of VGG-16 network structure adding a Batch Normalization layer before softmax layer, seen in Figure 10. A pre-trained VGG network with faces is available and provided by Oxford University [19].

Since this two proposed methods are related and using most of the times the same techniques, a comparison will be done to conclude which method works better and why. Always comparing with fine-tuning the entire output layer with all the classes seen so far. To conclude if it can be applied in face recognition stages.

As explained in next Sections, the main difference between these methods is that one store some representations to train when new classes comes in, and the second one do not store old samples to train when more classes are available.

3.1 Incremental Classifier and Representation Learning (iCaRL)

Lifelong learning systems are able to learn multiple tasks in an incremental way, as said before, the input has to be a stream of data in which examples of different classes occur at different times. Then in the output for every new data that comes in, the system has to be able to classify for the classes observed so far, and furthermore provide a competitive multi-class classifier. The computational requirements and memory footprint remain bounded with respect to the number of classes seen so far.

iCaRL use information of old data to train for every new batch of classes coming in, instead of taking the whole data set for the old classes, this proposed method select the samples that better represent the class by doing a nearest-mean class classification seen in Algorithm 1. So all the data for new classes and the selected exemplars for old classes are putted together in order to preserve the knowledge obtained before.

Let's say there is a feature function $\phi : X \leftarrow R^d$ and a classifier defined by $g_y(x) = \frac{1}{e^{w_i}}$ for each

```
**Input:** - $Y = \{y_1, \dots, x_t\}$ // Set of images for new classes.
- $\phi(x)$ // features for every sample
- $\mu_y$ // Set mean feature vectors (one for each class)
**Output:**
- $Z = \{y_1, y_2, \dots, y_t\}$ // Set of representations images for each class
for $i \leftarrow 1$ to $t$ do
    $\mu_i \leftarrow \frac{1}{|\{i:y_i\}|}\sum_{\{i:y_i\}}\phi(x_i)$ // Compute the average feature vectors
end
$Z_i* \leftarrow \underset{y \in Y}{argmin}|\phi(x) - \mu_y|$ // Select number of representations for each class
return $Z$
```

Algorithm 1: SELECT REPRESENTATIONS Nearest-Class-Mean (NCM) classifier

$y \in Y$ seen so far. Since the data set is a stream of data that comes in batches of classes, so there is more than one class for every batch which can be defined as $X^s, \dots, X^t$ where all representations in X^y are of class Y.

To train for the new batch of classes, the classification and the representation layer are trained, keeping the rest of the layers frozen. After the training process, the whole old data representation has to be

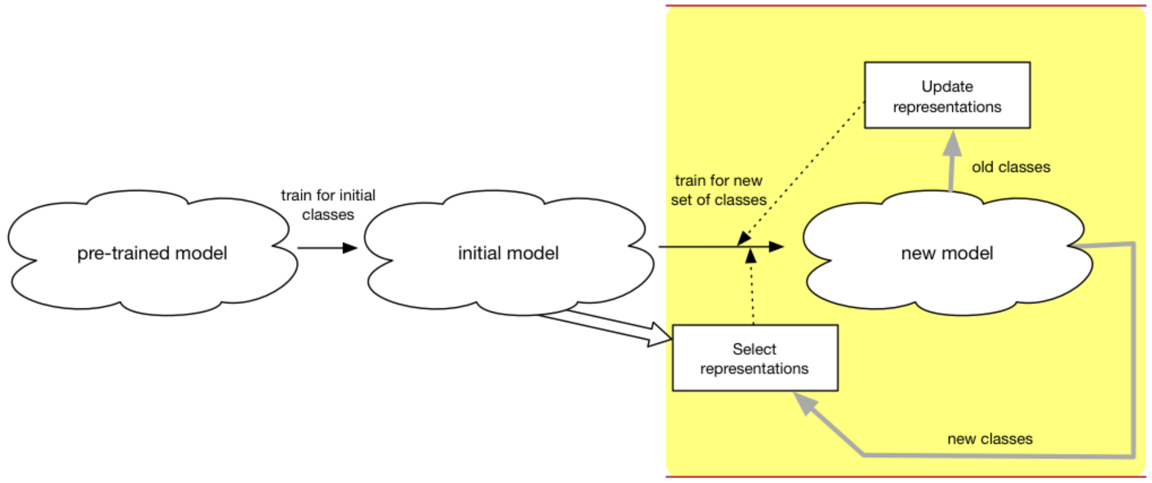


Figure 13: Schema of incremental training

recomputed by the actual feature function $\phi(x)$. Later has to re selected by the nearest-mean-class of exemplars. It takes more computational time and it consumes memory because the exemplar set is constantly growing in number of classes. To deal with it, a parameter that mark the total number of representations that can be stored is considered. The Nearest-Class Mean algorithm is exploited in this stage to take the most representative images for every class as seen in [4]1.

iCaRL relies on sets of exemplar images, it means that do not need any priory information about how many classes will occur. For every trained batch of classes, the classifier has to achieve competitive results. Some methods explained in last sections are used for training the model θ_s.

Principle of reharsal. To preform the update procedures for the model parameters for learning the

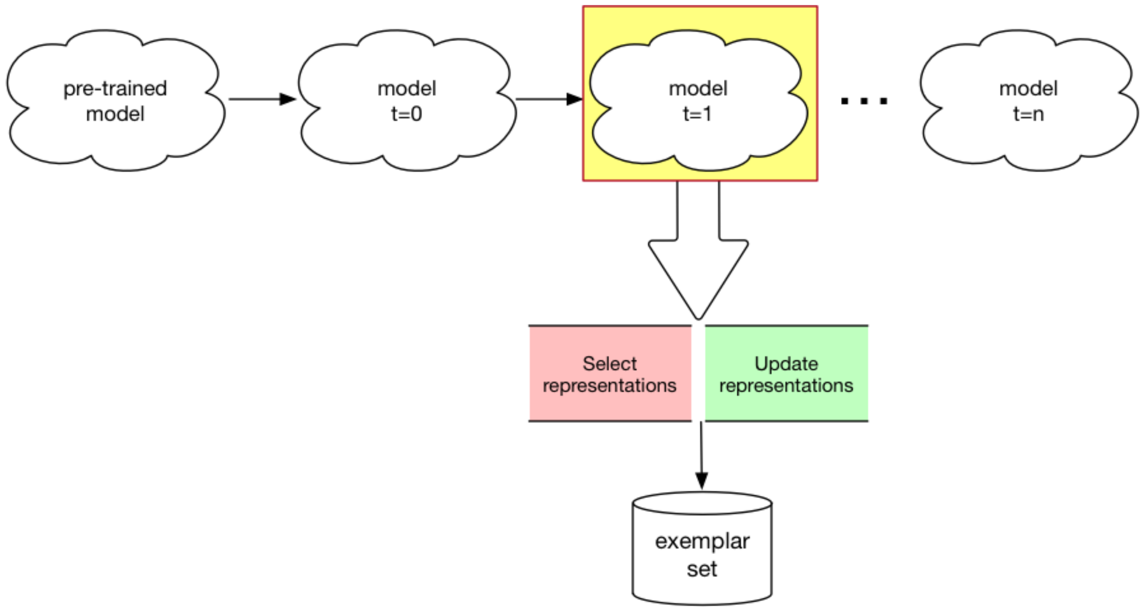


Figure 14: Schema of incremental training

representations, all the data of new representations is taken and also the data representations for the earlier classes.

Distillation Knowledge. This method for transfer learning between networks is also used in order to keep the knowledge for the old classes high while learning new classes at the same time. So the knowledge is not transferred to other network, is used to learn in the same network. The loss function called Knowledge Distillation loss found by Hinton [8] work well using the outputs of a network to well approximate the outputs of another.

Taking a look to the loss function proposed [20], the function is separated in two terms. One term for the classical classification loss using categorical cross entropy which represents the loss function on the learning for new classes, defined by $l(\theta) = -\sum_{i=1}^{s-1} \delta_{y=y_i} * log(g_y(x_i))$ where $\delta_{y=y_i}$ are the logits for the new classes and $g_y(x_i)$ are the predicted probabilities performed by the softmax classification layer.

The other term, has to preserve the knowledge for the old classes while the training for the new classes is performed 2.5.2 proposed by Hinton which says that you can define the loss function in a distillation knowledge stage by $l(\theta) = -\sum_{i=s}^{t} q_{y=y_i} * log(g_y(x_i))$ where $q_{y=y_i}$ are the outputs of the current model for the representations of the old classes which are called the exemplar representation sets. Distillation is well studied by Zhizhong Li and Derek Hoiem in [14] that uses it not to transfer the knowledge from one network to another but to transfer the knowledge learned before in the same network when new training data for new classes in available seen in Section 3.2.

In Algorithm 2 the principal steps for Class-Incremental Classifier Learning algorithm can be observed in different functions. Let's suppose that there is a model firstly trained with some classes, which some of the most representative images for each class are stored in exemplar sets. Then, some new data for training new classes appear, so in order to train the network the exemplar sets and the data for new classes has to be merged in *mergeSets* function making a training set. In order to finetune the network for new classes, some new outputs are added and initialized to zero, which is also included in *finetuneNetwork* function.

The implementation for training was simulated with streams of fixed batch classes. As it's explained, all the data of the new classes is used, instead of old data stored in exemplar sets, which representations are chosen by taking the most representative samples for each class seen so far.

Once the network is trained for the new classes, the exemplar sets has to be updated, so the features for all the representations of the exemplar sets and for the new representations has to be performed to later select the most representative samples for each class, included the new ones. After this two steps, the exemplar sets are stored. Figures 13, 14 shows that when the model changes, the exemplar set also changes. For old representations in exemplar sets, an update routine is called *updateRepresentations* while for new representations, a select routine is also called *selectRepresentations*.

The number of exemplar representations for each old class has to be chosen by defining a number of a total representations n_{total}, which is related to the memory available for the system and also of the total number of classes observed so far $n_{classes}$. So first there will be a lot of data for old classes, but when new classes come in and the total space n_{total} that was assigned to the exemplar sets, the number of representations has to be reduced by $n_{representations} = round(\frac{n_{total}}{n_{classes}})$, when $n_{representations}$ change, the exemplars stored for old classes has to be chosen again.

```
Input:
- $X = \{x_s, \ldots, x_t\}$ // Set of images for new classes.
- $Z = \{z_1, \ldots, z_{s-1}\}$ // Set of images representations for old classes.
- $L = \{l_s, \ldots, l_t\}$ //Labels for new classes
-$O = \{o_1, \ldots, o_{s-1}\}$ // Predictions for old classes
- $n_{respresentations}$ // Number of representations for class
- $\theta_s$ // Current model weights
Output:
- $Z = \{z_1, z_2, \ldots, z_t\}$ // Set of representations images for each class
- $\theta_t$ // Model weights
$Y, L \leftarrow mergeSets(X, Z, L, O)$// Merge image sets and labels vectors
$\theta_t, \phi_t \leftarrow finetuneNetwork(\theta_s, Y, L)$
$P \leftarrow computeFeatures(\phi_t, Y)$
$Z \leftarrow selectAndUpdateRepresentations(Y, P, n_{representations})$
$Z \leftarrow (z_1, \ldots, z_t)$
return $Z$
```

Algorithm 2: INCREMENTAL TRAINING

3.2 Learning Without Forgetting (LwF)

Another related work to deal with batches of classes presented by Li, Hoiem [14], the system is able to incrementally train a single network for learning multiple tasks, which makes this method a little bit different of the Incremental multi-class Representation learning explained in Section 3.1. The main difference between this two methods is that iCaRL requires learning one classifier for any input that can predict the observed classes, whilst this method introduced in this section, learns a separate classifier for each task and each classifier is evaluated only on the data from its own.

Learning without forgetting is heavily inspired in Less Forgetting Learning [10] which preserves the per-

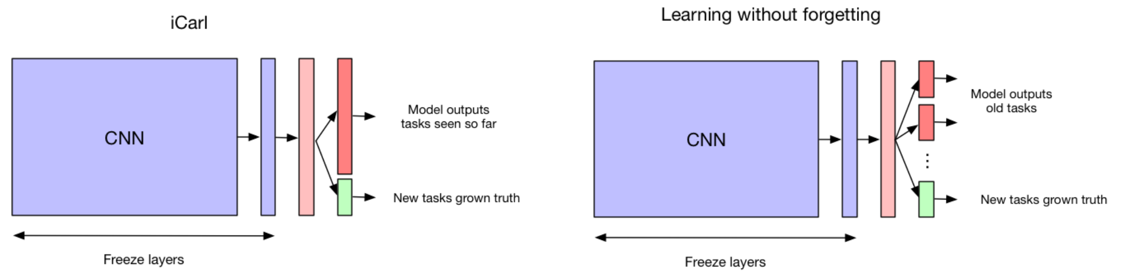


Figure 15: Proceature to train for new classes in both methods

formance for old tasks by preventing the shared representation to change. While Less Forgetting Learning method argues that task-specific decision boundaries should not change keeping the old tasks final layer unchanged, against Learning without Forgetting that discourages the old task output to change, and furthermore a jointly optimization is done for the shared representation and the final layer.

This approach rise with a combination of Distillation knowledge from networks while fine-tuning for each batch of classes. So once a task is learned, the training data does not need to be stored to future apply it for training as it's seen in Section 3.1 where some representations for the old classes are stored in order to preserve the knowledge.

Looking to Figure 16 it can be appreciated that when new classes are available, an output layer just for the new classes is finetunned while keeping the rest of the network frozen, which give the initialization for its weights and its connections to jointly train with the old output layer in order to take good results for both old and new classes. This procedure can be seen in Algorithm 3 provided in [14].

For each original task, the output probabilities for each image has to be close to the recorded output from the original network, and that's why the use of Knowledge Distillation loss to increase the weight for smaller probabilities, like explained in Section 2.5.2 and well-detailed [8].

It deals with modified versions of the recorded and current probabilities. To preserve the knowledge for old tasks, a loss balance weight λ_0 is applied to not overfit the network with new tasks.

```
Input:
- $\theta_s$ // Shared parameters.
- $\theta_o$ // task specific parameters for each old task
- $X_n, Y_n$ // training data and ground truth on the new task.
Output:
- $\theta_s^*, \theta_o^*, \theta_n^*$ // Updated parameters
$Y_o \leftarrow CNN(X_n, \theta_s, \theta_o)$// Compute output of old tasks for new data
$\theta_n \leftarrow RandIt(|\theta_n|)$
$Y_s \leftarrow CNN(X_n, \theta_s, \theta_o)$//Old task output
$Y_t \leftarrow CNN(X_n, \theta_s, \theta_n)$//New task output
$\theta_s^*, \theta_o^*, \theta_n^* \leftarrow argmin(\lambda_o L_{old}(Y_o, Y_s) + L_{new}(Y_n, Y_t) + R(\theta_s^*, \theta_o^*, \theta_n^*))$
return $\theta_s^*, \theta_o^*, \theta_n^*$
```

Algorithm 3: INCREMENTAL TRAINING (LWF)

3.3 Implementation

The network is trained using standard back propagation with mini batches of size 32 and a weight decay parameter of 0.008 for iCaRL method and 0.001 for LwF method. On the first batch of classes, when the model hasn't been trained yet, the learning rate is set at 1.0 in both methods because the networks do not have information of any class yet. It will outperform the results.

As suggested in [20] when doing the experiments, the learning rate has to be smaller than the experiments done in that work. A learning rate of 0.05 was chosen by trial and error experiments to find the value that outperform the results. Also, some regularization for the learning rate after each class of batches is considered in order to adapt the learning of the network for the new classes.

To perform the optimization in training, as typical in this kind of works, SGD is used because of its convergence rate at the different parameters such as the learning rate and the weight decay. this is used in a well-established network proposed by Oxford University in [22] that is VGG-16. Where its structure is described in Figure 10

A pre-trained VGG network for face recognition is available since 2015 when they train VGG-16 for a face recognition task in [19] which results seems to be more or less equal than the actual state-of-the-art for face recognition that is FaceNet [25] with an accuracy of 97.35% over Labeled Faces in the Wild dataset. To test it in an incremental learning stage, some modifications are done in top layers, as also is described in Figure 10 where the layers that are in blue blocks are frozen during training. The red block represents the finetunned layers. As it can be appreciate, the top layer was removed from VGGFace which contains over 1000 outputs and replaced to another one which contains the desired outputs for the number of classes of first batch.

To outperform the results originally obtained, a Batch Normalization layer was added before the softmax layer (fc8) as Figure 10 shows. As suggested in [20], adding a Batch Normalization layer, allow the network to keep the weights before the softmax layer in a range to work with. Sometimes the weights of the networks take high values that can saturate the activations, and Batch Normalization allows to keep these values in a understandable values for the network.

4 Experiments

The report for the results of this two implemented and tested methods for incremental class representation learning is done over a well-known face classification data set: FaceScrub [18] which basically is build by detecting faces in images returned from searches for public figures on Internet.

Lots of experiments designed to evaluate the performance of this two incremental learning methods [20, 14]. To later conclude which one of this approaches is the most effective method to learn new tasks while preserving the old ones as it's explained in Sections 2, 3. iCaRL method requires of training data for old classes, this data is selected in a representative way just taking the old samples data that are relevant considered by a distance with the mean feature vector of each class as Section 3.1 explains.

On the other hand, Learning without forgetting method do not require the old data to be stored for training for new classes, it means that when new classes are available LwF uses only images and labels for the new tasks.

The experiments are designed to evaluate whether iCaRL and LwF are effective methods to learn new tasks while preserving the performance for the old ones. This two methods are compared to fine-tuning, which gives the best model accuracy. The fine-tuning results can't be compared at all to the two incremental learning approaches exploited in this work, because of all the data set seen so far is used to train on every batch of classes, while the rest of the methods do not use all or any the samples for the old classes, focusing only to new classes samples.

4.1 Data set

The FaceScrub dataset was created using this approach [18], followed by manually checking and cleaning the results. There are a total of 106,863 face images of 530 celebrities, with about 200 images per person, which make it one of the largest public face dataset.

To build it, first the names are compiled of public figures from websites such as IMDb, also the genre is taken to later use to label training data for the gender classifier (which not apply in this work). Every name taken from public figures websites, is search in an image search engine such as Google, download the returned images to apply the face detector to extract potential faces for each person.

4.2 Results

The comparison between the different methods are done over a stream of data, it means that the data set add cumulatively new tasks to the system to simulate a scenario in which new faces in our case are gradually added to the prediction vocabulary.

As it's said we experiment on gradually adding FaceScrub tasks, which are taken from the cropped faces version of the data set, since the cropped version for its identities faces is done.

Figure 16: FaceScrub data set

We can differentiate this two different methods that are being analyzed, as Section 3 explains, by analyzing what we do during training. These methods are really similar but iCaRL in our case outperform the continuous learning scenario proposed [20] for object recognition using CIFAR-100. It seems that depends on the data set this methods work more or less better, if we compare object recognition and face recognition stages and data sets. That's why the importance of the data set is explained in Section 4.2.1.

An analysis with different initialization methods is done over the weights of the new layer to add new capabilities to the system. In iCaRL method, two different initialization are suggested, and the conclusion was that the system works better with zeros initialization instead of randomly initialization. Otherwise, LwF method seems to outperform when the initialization is done by fine-tuning only the new added layer to use it as initialization to later joint train for old and new tasks as Algorithm **??** shows.

For iCaRL method, some experiments are done in order to contrast the results to LwF and fine-tuning methods. Since iCaRL uses a sets of exemplars for the old classes it's quite interesting to take a look in how many representations are needed in order to have competitive classification models after training for every batch of classes. First, we didn't consider a total size for memory to store old exemplars, analyzing the number of representations needed. The results in Figure **??** shows 3 curves for iCaRL accuracy performance which corresponds to a number of representations of 5, 10 and 20 respectively. As it's shown there is not much differences between taking 10 or 20 representations per class, which curve is similar to fine-tuning method.

As suggested in [20], to implement the system in real situations, a total memory size has to be chosen since the system will grow every time new classes are available. It means that if the total number of representations of the classes seen so far exceeds a certain chosen parameter n_{total}, the representations for the exemplar sets has to be re chosen by keeping a number of representations that do not exceed the total number of representations. In this case, it can be appreciated that the system has limitations over the number of classes seen so far and its number of representations stored for each class. In other words, the system is not able to grow it's tasks to *"infinite"*.

4.2.1 Importance of the data set

As observed in 3.1 compared to 3.2 and proved in Figure 17, iCaRL method is always achieving better results than the other method. Comparing the results obtained using iCaRL method in face recognition to the other ones published before [14, 10, 11] which proposes are mainly focusing on object recognition tasks, it can be appreciated the importance of having a large enough dataset to feed the incremental network.

To prove it, LwF method implemented by the author is achieving better results as iCaRL experiments with LwF does [20] which achieve around 20% after 100 classes seen so far coming in batches of 10 classes, and around of 50% using iCaRL method. Looking at Figure 17 we can see that using more rich data sets like FaceScrub that give us good quality images will outperform this incremental learning methods. Instead, [20] used CIFAR-100 data set which contains images divided in 100 classes for object recognition, every image have a size of 32x32 which is considered a low quality images.

4.2.2 Importance of the exemplar sets

As it's seen in Section 3 comparing [20] and [14] and contrasted in Figure 17, iCaRL is achieving far enough better results than LwF just for using some representative samples for each class.

As suggested in [20] a total number of representations is chosen n_{total}, which accuracy curves can be seen in Figure 17 as *iCaRL adaptive*. This experiments to adapt the total number of representations for the exemplar sets are done over a maximum of 500 representations for all classes, with a initial number of 20 represenions for each class. For batches of 10 classes and looking from 10 to 100 classes so far the result show that the curve is under 20 representations curve after the maximum number of representations is achieved. Also looking for batches of 50 classes, the maximum number of representations are achieved earlier and that's why its worse results but still performing better than LwF.

We can extract conclusions also looking at the Figures 18 and 19 where on the top side of both figures, the accuracy for old and new classes is shown during training. It can be appreciable that if a higher number of representations for old classes in exemplar set is augmented, then the two curves gather together.

If we analyze the adaptative method to chose the number of representations for each class in expemplar sets it can be appreciated that the curves for old and new classes are more or less joined, but when the total number of representations is achieved, the accuracy for old classes is reduced and the accuracy for new classes is heavily augmented

In LwF, the results are quite different because of the absence of the exemplar set. As it's explained in Section 3.2, this method is not using old representations anymore and this is reflected to the curves shown in Figures 17, 18 and 19. Comparing the accuracy for old classes and new classes in every batch we can appreciate that LwF performance is better using a high batch class size. The results shows that LwF results are better when batches of 50 classes are used. Instead of iCaRL method where the performance is better when low number of classes are coming in every batch, but still performing better results than LwF in higher batch class sizes.

4.2.3 Choosing "good" training parameters

The most important task performing the experiments is the correct chose of the hyperparameters, which will change the results heavily when doing bad practices.

As said in [14] in the effect of lower learning rate of shared parameters which results seems that lowering the learning rate does not prevent fine-tuning from significantly reducing original tasks performance, *"Simply Reducing the learning rate of shared layers is insufficient for original task preservation"*.

To outperform LwF with FaceScrub dataset, an initial high learning rate of 1.0 was chosen, in order to later preserve the original tasks knowledge, in other words, train more for old tasks than for new ones.

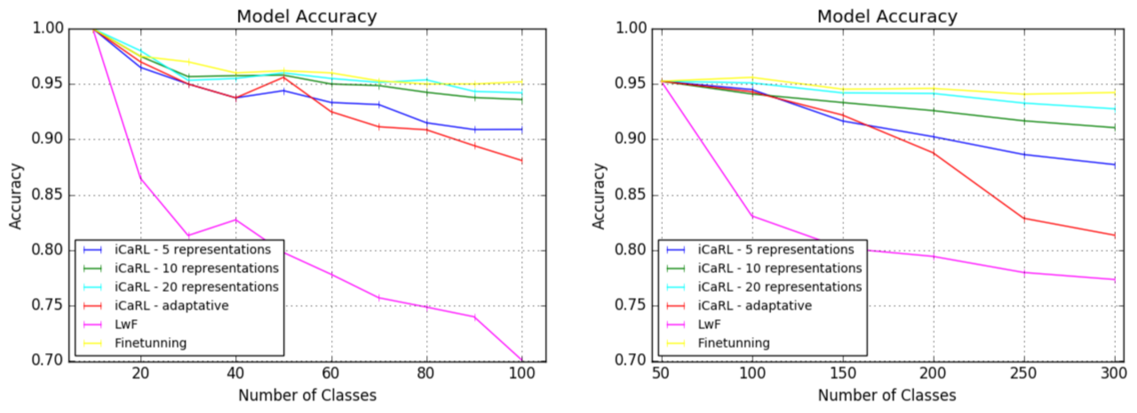


Figure 17: Class Incremental Training.

Tested with FaceScrub data set with 10 (left-side) and 50 (right-side) classes per batch

After the initial model is trained, the learning rate can be reduced. This won't get good results for new tasks at the moment but the performance will be adapted later when new classes are coming. As it's seen in Figure 20 comparing performance of iCaRL and LwF during training in two different class batch training stages.

Some trial and error experiments were done to adjust parameters like learning rate, decay, batch size and the number of epochs. First, a high learning rate of 1.0 is used followed by a decay of 0.1 in 30 epochs for the training of the initial model (when first batch class comes in). The parameters chosen initially are the same for iCaRL and LwF.

To contrast firstly learned classes with the new ones, another parameters were chosen in order to preserve as maximum as possible the knowledge for the old classes. For LwF the learning rate is 0.04, which is considerably low compared to the initial one, followed by a decay of 0.008 with a number of 70 epochs. LwF seems to work better using more epochs, always dealing with the preservation of knowledge.

For iCaRL a learning rate of 0.05 is chosen followed by a decay of 0.008 with 20 epochs, which make also iCaRL faster in training stage. Figure 20 shows that iCaRL training is more stable than LwF, and it's just because of the usage of the old representations.

To initialize the new nodes added when new classes are available for training was also chosen by a trial and error experiments. For LwF and as it's suggested in [14] the initialization is done by fine-tuning only the output new layer and use the weights as the initialization to joint train with old outputs. iCaRL initialization weights and connections are chosen to zero which seems to work better than random initialization.

We can conclude saying that this method perform good results if the representations of exemplar sets are fixed instead of changing the representations if a maximum number of representations is achieved, which performs bad results in comparision but stull performing better than LwF as Figure 17 shows

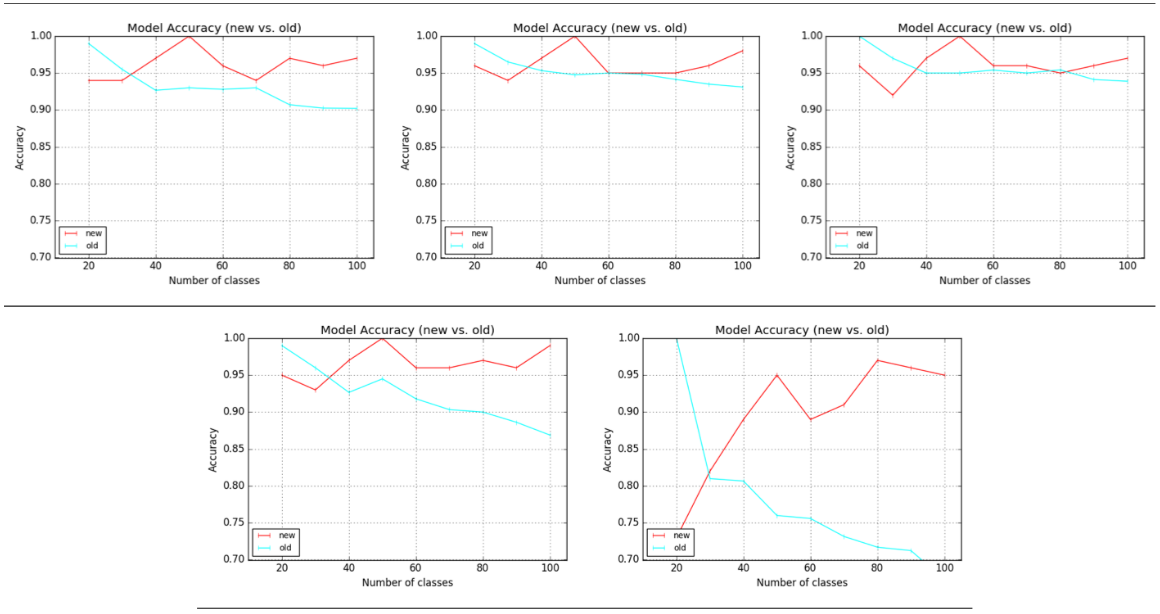


Figure 18: Class Incremental Training accuracy test for old and new classes (10 classes per batch).

Tested with FaceScrub data set with 5 (top left), 10 (top middle), 20 (top right), adaptative (bottom left) number of representations and LwF (bottom right)

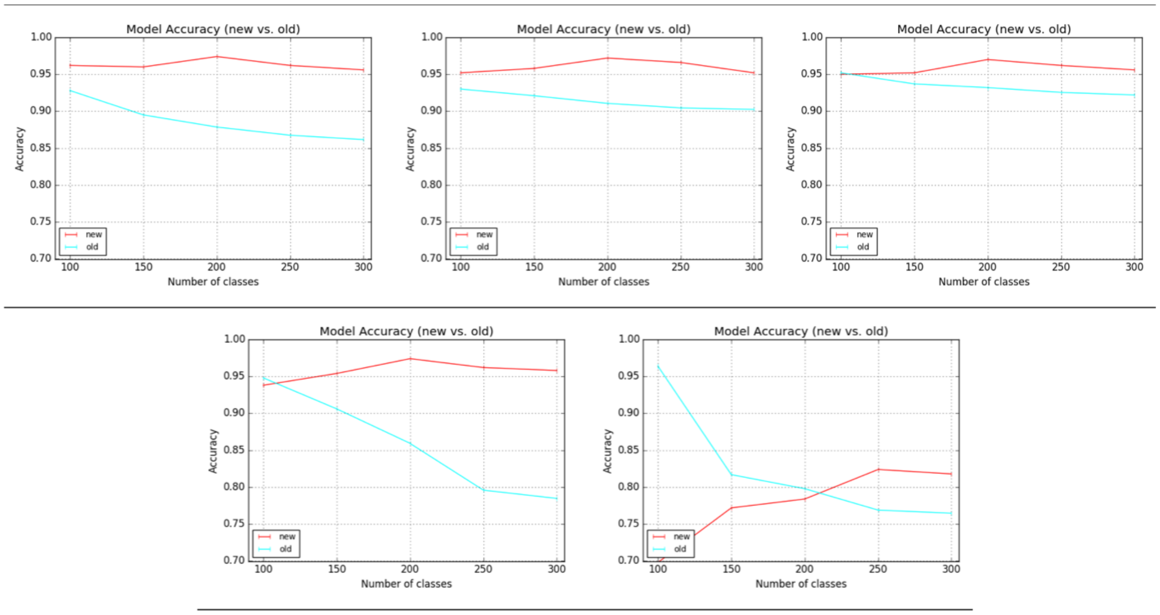


Figure 19: Class Incremental Training Accuracy test for old and new classes (50 classes per batch)

Tested with FaceScrub data set with 5 (top left), 10 (top middle), 20 (top right), adaptative (bottom left) number of representations and LwF (bottom right)

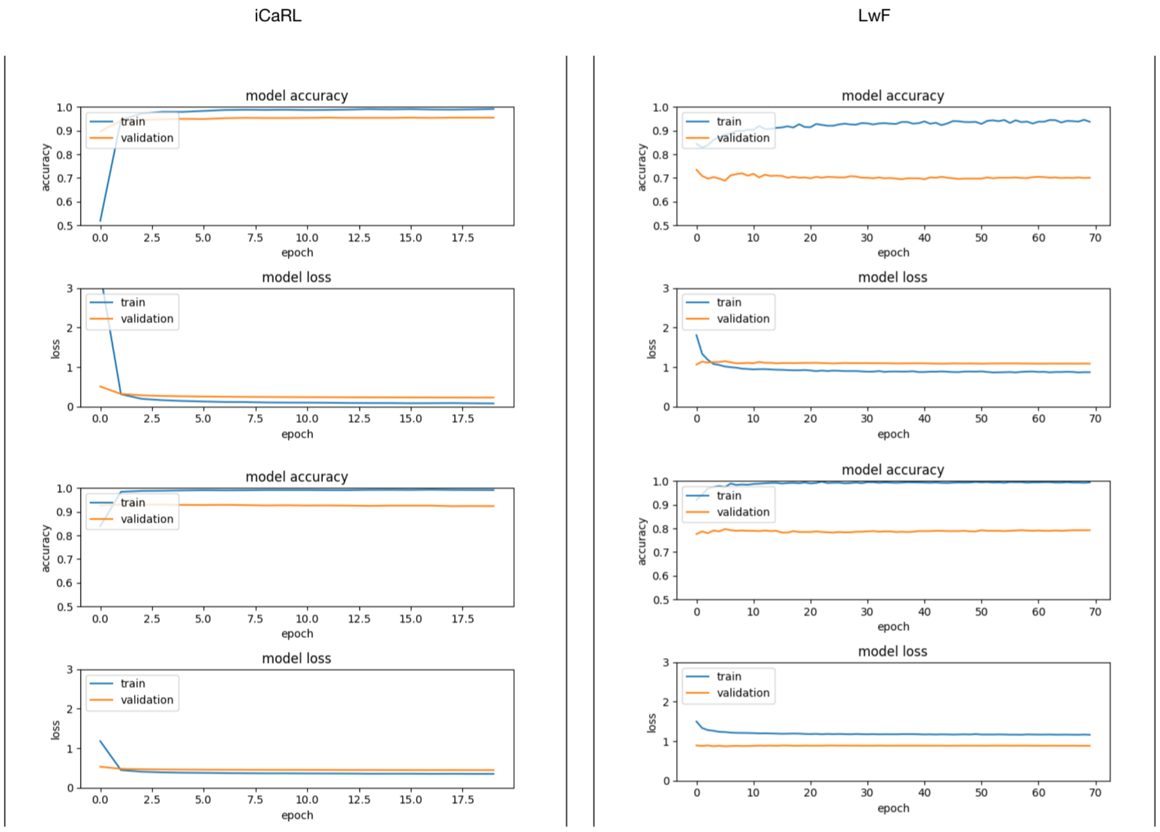


Figure 20: Class Incremental Training evolution accuracy and loss.

Trained with FaceScrub data set in second batch of classes (top) and 5th batch of classes (bottom) for both methods

5 Budget

This project is carried out using the Development Platform server from Image Processing Group (GPI) from UPC, which make possible the execution of the experiments and test. All libraries used are open source for non-commercial profit licence.

The project started at February 2017 and ends in July 2017, what summe a total of 5 months. The dedication proposed by the project supervisors was to work 20 hours per week with weekly meetings with supervisors in order to make a following to the project.

Two supervisors have assisted to the realization of this project, as it's said, there where weekly meetings of a duration of 1 hour. The supervisors fee is about 55€/hour. The author's salary as a junior engineer is around 2000€/month working full time. Since this project only require 20 hours of work per week the author's salary is around 1000€/month.

Finally, if the server was outsourced to a company like Google, which provide access to their servers using Google Computing Cloud engine. Analyzing the needed hardware shown in Table 1 we can approximate the costs of its utilization by a total of 1.237,28$/month. Table 2 shows all the information about the cost of this project that would be around 13.216,60€.

Hardware	Model	Properties
Graphics Unit Processor (GPU)	NVIDIA TESLA k80	1 GPU 12 GB
CPU	-	60 GB 16 Kernels
Storage	-	500 GB

Table 1: Hardware Needed

Resources	Cost/month	Months	Cost
Junior Engineer	1000€	5	5000€
Supervisor 1	220€	5	1100€
Supervisor 2	220€	5	1100€
Server	1.203,28€	5	6016,6 €
		Total:	13216,6€

Table 2: Costs of the project

6 Conclusions and Future Work

The main motivation about this kind of research and its propose it's to move Artificial Intelligence community beyond learning algorithms that considers the nature of systems that are able to learning over a lifetime. That's why the research is always to improve the limitations of the actual state-of-the-art proposes published so far.

In this thesis it is shown how to use convolutional learned features extracted from VGG-16 to work in a incremental way. Growing it number of classes using some proposed novel methods which are achieving really good results on face recognition.

This work proposes a novel method published in the last year, but we have use it to solve the task of face classification in an incremental way. The time consumption for training of large-scale data sets as FaceScrub is a problem under constant research. The incremental learning methods analyzed in this work are trying to solve it.

The only limitations that can be appreciated for iCaRL is the total number of representations stored for preserving knowledge for the old classes. If the data set has an unlimited amount of data as in online learning stages, this method performance will be limited to the maximum number of representations in exemplar sets chosen.

Based on this work analysis, we conclude that incremental learning can be directly achieved just by fine-tuning. We use fine-tuning basis to train a system that learn and retain knowledge at the same time. It's interesting the scalablity property of this systems, able to scale up a large number of training examples.

The usage of a data set with cropped faces help to outperform the results. Also the iCaRL strategy for learning the classifiers and the feature representations simultaneously. Observing that the quality of the images is an important thing to take into account while using this method.

For future work try other architectures based on the combination of this two methods and its extensions. Also, making the system robust to noisy samples since the chosen data set is a *"clean"* one.

On the other hand, the computational time taken from training the system by a incremental way compared to train it with all training examples is not improved by this methods.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF : Speeded Up Robust Features.

[2] H. Bekel, I. Bax, and G. Heidemann. Adaptive computer vision: Online learning for object recognition. *Proc. Pattern Recognition Symposium (DAGM)*, 3175 of LN:447–454, 2004.

[3] Mirza Cilimkovic. Neural Networks and Back Propagation Algorithm.

[4] Piew Datta and Dennis F Kibler. Symbolic Nearest Mean Classifiers. *Association for the Advancement of Artificial Intelligence*, pages 82–87, 1997.

[5] Jia Deng, Wei Dong, Richard Socher, Li-jia Li, Kai Li, and Li Fei-fei. ImageNet : A Large-Scale Hierarchical Image Database. pages 2–9.

[6] Sachin Sudhakar Farfade, Mohammad Saberian, and Li-Jia Li. Multi-view Face Detection Using Deep Convolutional Neural Networks. 2015.

[7] Motonobu Hattori. Avoiding catastrophic forgetting by a dual-network memory model using a chaotic neural network. *World Academy of Science Engineering and . . .*, (36):853–857, 2009.

[8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. pages 1–9, 2015.

[9] Geoffrey E Hinton and Simon Osindero. A fast learning algorithm for deep belief nets âĹŮ 500 units 500 units. 2006.

[10] Heechul Jung, Jeongwoo Ju, Minju Jung, and Junmo Kim. Less-forgetting Learning in Deep Neural Networks. XX(X):1–5, 2016.

[11] Christoph Käding, Erik Rodner, Alexander Freytag, and Joachim Denzler. Fine-tuning Deep Neural Networks in Continuous Learning Scenarios. *ACCV 2016 Workshop on Interpretation and Visualization of Deep Neural Nets*, 2016.

[12] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. 2016.

[13] P. LeCun, Yann; Bottou, L.;Bengio, Y.;Haffner. Gradient-Based Learning Applied to Document Recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[14] Zhizhong Li and Derek Hoiem. Learning without forgetting. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9908 LNCS:614–629, 2016.

[15] David G Lowe. Distinctive Image Features from Scale-Invariant Keypoints. pages 1–28, 2004.

[16] Vijayshree More and Abhay Wagh. Improved Fisher Face Approach for Human Recognition System using Facial Biometrics. 2(2):135–139, 2012.

[17] Hyeonseob Nam and Bohyung Han. Learning Multi-Domain Convolutional Neural Networks for Visual Tracking. 2015.

[18] Hong Wei Ng and Stefan Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE International Conference on Image Processing, ICIP 2014*, pages 343–347, 2014.

[19] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep Face Recognition. *Procedings of the British Machine Vision Conference 2015*, (Section 3):41.1–41.12, 2015.

[20] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert. iCaRL: Incremental Classifier and Representation Learning. 2016.

[21] Daniel L Silver, Qiang Yang, and Lianghao Li. Lifelong Machine Learning Systems : Beyond Learning Algorithms. *AAAI Spring Symposium Series*, (Solomonoff 1989):49–55, 2013.

[22] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *International Conference on Learning Representations (ICRL)*, pages 1–14, 2015.

[23] Marijeta Slavkovi and Dubravka Jevti. Face Recognition Using Eigenface Approach *. 9(1):121–130, 2012.

[24] Ilya Sutskever. Sequence to Sequence Learning with Neural Networks. pages 1–9.

[25] Y Taigman, M Yang, and M.A. Ranzato. Deepface: Closing the gap to humal-level performance in face verification. *CVPR IEEE Conference*, pages 1701–1708, 2014.

[26] Donald R Tveter and Commercial Use. The Backprop Algorithm. *Network*, 1995.

[27] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, 1:I–511–I–518, 2001.

[28] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. *Neural Networks*, 16(10):1429–1451, 2003.

[29] Tianjun Xiao, Jiaxing Zhang, Kuiyuan Yang, Yuxin Peng, and Zheng Zhang. Error-Driven Incremental Learning in Deep Convolutional Neural Network for Large-Scale Image Classification. *MM '14 Proceedings of the ACM International Conference on Multime*, d:177–186, 2014.

[30] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014.