

USING CNNs IN THE DOMAIN OF VISUAL TRADEMARK RETREIVAL

A Degree Thesis

**Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona**

Universitat Politècnica de Catalunya

**and developed in the
Institute of Information Technology (ITEC)**

Alpen-Adria Universität, Klagenfurt

by

Al·lodi Jutglà Serrat

**In partial fulfilment
of the requirements for the degree in
AUDIOVISUAL SYSTEMS ENGINEERING**

AAU advisor: Mathias Lux

UPC advisor: Xavier Giró

Klagenfurt, March 2018



Abstract

Nowadays we are immersed in the age of Artificial Intelligence and it seems that every application has to be developed following this tendency. But even this actual boom, this technique is not so common to find in the domain of image trademark retrieval. Thus, what this thesis proposes is to create a tool to help the people in charge of the process of indexing and classifying the upcoming visual trademarks, to support them with suggestions following a standard classification.

The project involves all the process end-to-end of a classification task in deep learning. From downloading and extraction of a data set for training and testing, passing through the analysis of the state of the art and finishing with the evaluation of the results in our own database.

As this field is constantly developing we cannot predict what the future will bring. However, using deep learning in this scenario may help in the future to have more concise labelling and classification of visual trademarks compared to the results of the manual process.

Resum

Avui en dia ens trobem immerses en l'era de la intel·ligència artificial i sembla que totes les aplicacions hagin de ser desenvolupades seguint aquesta tendència. Però tot i l'actual boom en que vivim, aquesta tècnica no es gaire comú trobar-la en el domini de la retribució d'imatges de marques. Per tant, el que proposa aquest treball és crear una eina per ajudar les persones que s'encarreguen del procés d'indexar i classificar les noves marques, recolzant-los amb suggeriments per seguir una classificació estàndard.

El projecte involucra tot el procés de principi a fi d'una tasca de classificació en aprenentatge profund. Des de la descàrrega i extracció d'un conjunt de dades d'entrenament i test, passant per l'anàlisi de l'estat de l'art i acabant per l'avaluació dels resultats en la nostra pròpia base de dades.

Ja que aquest camp està en constant desenvolupament, no podem predir el que ens depararà el futur. Tot i així, usant l'aprenentatge profund en aquest escenari, pot ajudar en un futur per tenir una tasca d'etiquetatge i classificació de marques comercials comparable als resultats del procés manual.

Resumen

Hoy en día no encontramos inmersos en la era de la inteligencia artificial y parece que todas las aplicaciones hayan de ser desarrolladas siguiendo esta tendencia. Pero pese al actual boom en el que vivimos, esta técnica no es muy común encontrarla en el dominio de la retribución de imágenes de marcas. Por lo consiguiente, este trabajo propone crear una herramienta para ayudar a las personas que se encargan del proceso de indexación y clasificación de nuevas marcas, apoyándolos con sugerencias para seguir una clasificación estándar.

El proyecto involucra todo el proceso de inicio a final de una tarea de clasificación de aprendizaje profundo. Yendo desde la descarga y extracción de conjunto de datos de entrenamiento y test, pasando por el análisis del estado del arte, y acabando por la evaluación de los resultados en nuestra propia base de datos.

Ya que este campo está en constante desarrollo, no podemos predecir qué nos deparará el futuro. Aún y así, utilizando el aprendizaje profundo en este escenario, puede ayudar en un futuro a tener una tarea de etiquetado y clasificación de marcas comerciales comparables a los resultados del proceso manual.



Acknowledgements

I would like to recognize the work done by my principal advisor Mathias Lux and my co-advisor Xavier Giró because without them this project would not be possible.

I would also want to show my gratitude to Alpen-Adria Universität and especially to the Institute of Information Technology (ITEC) for providing all the necessary equipment and a place to develop my project.

In addition, I want to distinguish the facilities that the project OpenData from the Oficina Española de Patentes y Marcas has given to me. Without the open data of distinctive trademarks provided by them could be impossible to develop the thesis.

Revision History and Approval Record

Revision	Date	Purpose
0.0	07/02/2018	Document creation
0.1	08/02/2018	Document revision
0.2	09/02/2018	Document revision
0.3	17/02/2018	Document revision
1.0	23/02/2018	Document revision
1.1	26/02/2018	Document revision
1.2	05/03/2018	Document revision
2.0	26/03/2018	Final revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Al·lodi Jutglà Serrat	alodi.jutgla@alu-etsetb.upc.edu
Mathias Lux	mlux@itec.aau.at
Xavier Giró Nieto	xavier.giro@upc.edu

Written by:		Reviewed and approved by:	
Date	26/03/2018	Date	25/03/2018
Name	Al·lodi Jutglà Serrat	Name	Mathias Lux Xavier Giró
Position	Project Author	Position	Project Supervisor

Table of contents

Abstract	1
Resum.....	2
Resumen.....	3
Acknowledgements	4
Revision History and Approval Record	5
Table of contents.....	6
List of Figures	8
List of Tables:.....	9
1. Introduction.....	10
1.1. Project Overview and Goals	10
1.2. Requirements and Specifications.....	11
1.3. Work Plan	12
1.3.1. Work Breakdown Structure.....	12
1.3.2. Work Packages, Tasks and Milestones.....	13
1.3.3. Work plan modifications and Incidences	17
1.3.4. Time Plan (Gantt Diagram).....	18
2. State of the art of the technology used or applied in this thesis:.....	19
2.1. Evolution of CNNs.....	19
2.2. From CPU to GPU	20
3. Methodology and project development:	22
3.1. Set up the environment	22
3.2. Database.....	22
3.2.1. WIPO and Vienna Classification	23
3.2.2. Data extraction.....	24
3.2.3. Working Datasets	25
3.3. CNNs, the definitive algorithm.....	26
3.3.1. Overview of a CNN	26
3.3.2. ConvNet from scratch	28
3.3.3. Pre-trained CNN	28
3.3.4. Training and Evaluation Process.....	32
4. Results	34



4.1.	ConvNet from scratch	34
4.1.1.	Simplest first model (1)	34
4.1.2.	Update of the first model (2)	35
4.2.	Pre-trained VGG16Net	36
4.2.1.	VGG16Net with First Dataset (3)	36
4.2.2.	VGG16 applying fine tuning (4)	37
4.3.	Inception-ResNet (5)	39
4.4.	Qualitative evaluation of the results.....	41
5.	Budget	45
6.	Environmental Impact.....	46
7.	Conclusions and future development:	47
	Bibliography:	48
	Glossary	49

List of Figures

Figure 1 – Evolution of deep learning [Source: alltechbuzz.net]	10
Figure 2 – Systematic Deep Learning Process	13
Figure 3 – Project Proposal Gantt Diagram.....	18
Figure 4 – Final Report Gantt Diagram.....	18
Figure 5 – CNNs Evolution	20
Figure 6 – CPU vs GPU processing speed [Source: Nvidia’s Presentation]	21
Figure 7- Local patterns division [Source: Deep Learning with Python]	26
Figure 8 – Some sigmoid functions compared [Source: wikipedia.org].....	27
Figure 9 – ReLU function [Source: researchgate.net]	27
Figure 10 – Architecture of a typical CNN [Source: ujjwalkarn.me].....	28
Figure 11 – Left side a real image of a dog[Source]; Right side the abstract logo based on the image[Source]	29
Figure 12 – Architectures of VGG Net; D produces the best results[Source: kdnuggets.com].....	29
Figure 13 – Number of parameters of the configurations (in millions)	29
Figure 14 – Basic Inception module [Source: Going deeper with convolutions]	30
Figure 15 – GoogLeNet’s architecture [Source: hacktilldawn.com].....	30
Figure 16 – Residual learning block [Source: Deep Residual Learning for Image Recognition].....	31
Figure 17 – Schematic diagram of Inception-ResNet [Source: research.googleblog.com].....	31
Figure 18 – Neuron model [Source: tensorfly.cn]	33
Figure 19 – Architecture of the first model.....	34
Figure 20 – Accuracy and loss of the first model	35
Figure 21 – Architecture of the update of the first model.....	35
Figure 22 – Accuracy and loss of the update of the first model	36
Figure 23 – Architecture of the VGG16 model adapted to our task	36
Figure 24 – Accuracy and loss of the VGG16 with First Dataset	37
Figure 25 – Last layers of VGG16 Net.....	38
Figure 26 – VGG model applied fine tuning	38
Figure 27 – Accuracy and loss VGG16 applied fine tuning.....	38
Figure 28 – Last layers of Inception-ResNet.....	39
Figure 29 – Inception-ResNet model adapted to our task	40
Figure 30 – Accuracy and loss Inception-ResNet model	40

List of Tables:

Table 1 – Work Package 1	14
Table 2 – Work package 2	14
Table 3 – Work Package 3	15
Table 4 – Work Package 4	15
Table 5 – Work Package 5	16
Table 6 – Work Package 6	16
Table 7 – Milestone list	17
Table 8 – Vienna Classification Categories.....	23
Table 9 – Predicted percentage of 1st image.....	41
Table 10 – Predicted percentage of 2 nd image.....	42
Table 11 – Predicted percentage of 3 rd image	42
Table 12 - Predicted percentage of 4 th image	43
Table 13 - Predicted percentage of 5 th image	44
Table 14 – Theoretical personal cost	45

1. Introduction

Nowadays it seems that we are inside the age of Artificial Intelligence (AI), more concretely inside the Machine Learning and Deep Learning age, and it will be the new industrial revolution of the 21st century. But despite the multiple applications which use this technology, it's not really common to find it in the domain of image trademark retrieval. Thus, in this thesis proposed and supervised by Mathias Lux and Xavier Giró, we applied the CNNs in order to classify visual trademarks. This is typically a process done by hand by trained annotators, which is well controlled by a common taxonomy -- the Vienna Classification System -- but very inconsistent due to the large number of people and trademark offices involved.

1.1. Project Overview and Goals

As we can see all around, even when we are not really conscious of it, we are surrounded by applications that are powered by Artificial Intelligence. In our daily life, from the first moment of the day, which we check in our smartphone the state of the traffic to go to work, to the complexity of an autopilot that some commercial flights use, we are using Artificial Intelligence.

As Max Tegmark, professor at MIT, says in his book *"Life 3.0: Being Human in the Age of Artificial Intelligence"*, Artificial Intelligence is affecting crime, wars, justice, jobs, society and our sense of being human.

As you can see in Figure 1, Artificial Intelligence is the big bubble which gives a name to the entire discipline, but inside it, there's the denominated Machine Learning which includes the technique of Deep Learning.

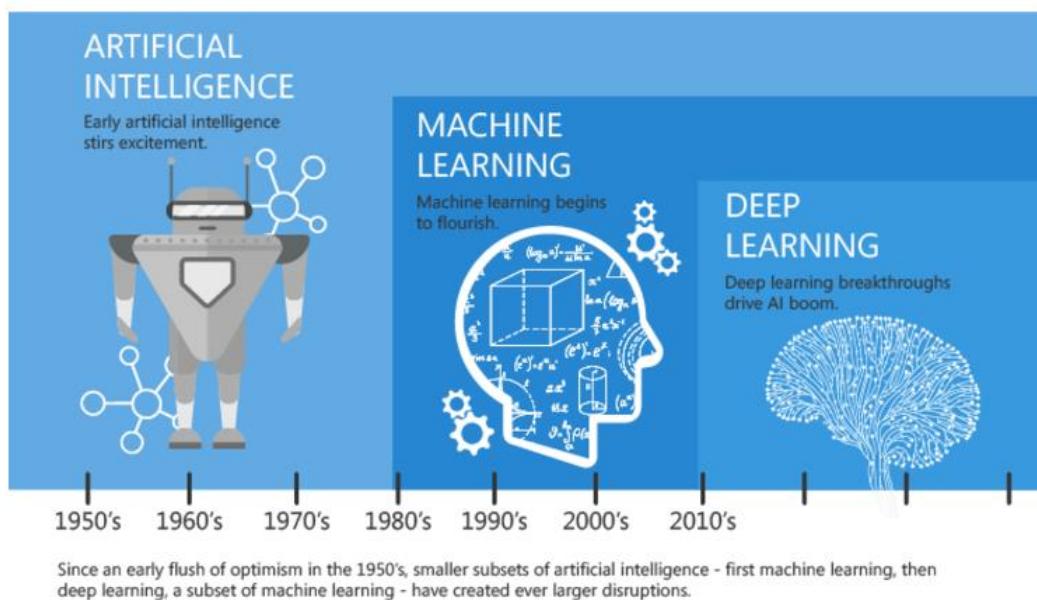


Figure 1 – Evolution of deep learning [Source: alltechbuzz.net]

Taking advantage of this boom of the technology and the resulting availability of ready-to-use tools and frameworks, in this project we wanted to apply the benefits of Deep Learning technique to visual trademark classification. This can be considered as a typical computer vision classification problem.

Our approach is based on the need for classification of newly registered visual trademarks. Each trademark submitted for registration at a patent and/or trademark office has to be classified manually using the Vienna Classification system, a standardized set of classes that can describe visual trademarks. The process is largely done manually and due to the large number of offices and people involved quality of the annotations varies over time and organizational unit. Our aim is to use deep learning to provide a baseline classification and support annotators in their daily work.

In need of training and test data we used the open data of trademark images provided by the Oficina Española de Patentes y Marcas (OEPM), which was already classified with the categorization of the Vienna Classification. We have applied and evaluated Deep Learning techniques, more specifically Convolutional Neural Networks (CNNs), to determine appropriate categories automatically. . The aim of this classification is to have an accurate register of every brand and then compare easily the similarity between trademark logos.

Consequently, the main goals of the project defined in the Project Proposal were:

- 1- To find a trademark database which is already classified in order to have enough data to train, validate and test our neural network.
- 2- To evaluate appropriate types of CNNs and pre-trained models for our case. Many of them deal with real life images in contrast to highly abstract visualizations, which are often the case with logos and visual trademarks.
- 3- To create a new CNN with the proper architecture and layers or to modify an existing CNN pre-trained with other datasets.
- 4- To achieve the best results possible, thus we can use them to annotate trademark logos which have not been tagged before, i.e. new visual trademarks or those without annotations.
- 5- To develop the necessary documentation in order to have the trace of the project.

1.2. Requirements and Specifications

Project requirements:

- Given a new dataset of logos or just a new brand to classify, the program must be able to classify these logos in the category that really belongs.
- The test result should give a good accuracy in order to know if this logo can be similar or not to another one previously created.

Project specifications:

- The programming environment will be Keras, which is a high-level neural networks API, written in Python and running in TensorFlow. It has been chosen because of the support

for easy and fast prototyping, its support for CNNs and the possibility to run the CNNs on the GPU.

- To have a good result the program needs a large dataset of logos in order to train as much as we can the CNN. At least more than 10 different classes with more than 5.000 logos each category.
- In order to evaluate the results, the dataset has to be split into three parts: the training part of the set in order to train the CNN, then a validation set to evaluate the quality of the trained model. Finally, the test set that has to be independent of the training part in order to evaluate and compute the results.

1.3. Work Plan

1.3.1. Work Breakdown Structure

The breakdown structure of this project follows the basic structure of a classification problem in the domain of Machine Learning and Deep Learning. But in this case, we have given more importance in the preparation of the database in order to collect a big amount of data to achieve optimum results. Then, we have another important part of improving the results. As we work with CNNs, we need different tunings in order to accomplish decent results.

At the same time as we work with the database, we have to set-up the working environment with the needed software. In this case the environment is Keras, based on Tensorflow and running on NVIDIA CUDA in an Ubuntu 16.04 OS.

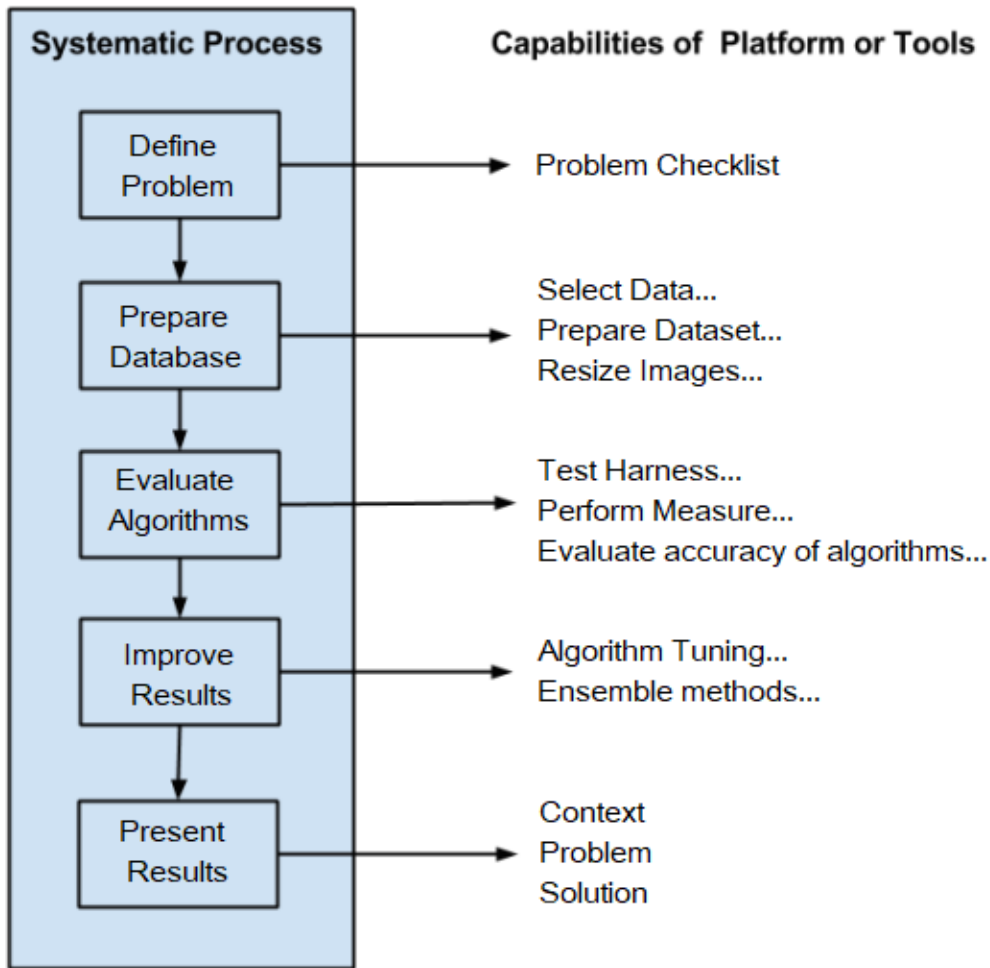


Figure 2 – Systematic Deep Learning Process

Thus, the project plan will follow Figure 2 diagram. Once the problem is well defined, we have to prepare the database selecting accurately the data and preparing it to accomplish the input specifications of the system. Then evaluate, test and improve the results of the algorithms in order to achieve the main goals. Finally, synthesize the extracted results and present in the best way possible. In the next points, it is exposed an explanation of all this process.

1.3.2. Work Packages, Tasks and Milestones

Work Packages:

Project: Define Problem	WP ref: (WP1)
Major constituent: Documentation	Sheet 1 of 1
Short description: Describe the issues that need to be solved and investigate the state of the art in order to approach better the solution.	Planned start date: 02/10/2017
	Planned end date: 30/10/2017
	Start event: Project assignment End event: Project Proposal and

	Work Plan	
<p>Internal task T1: Problem Description</p> <p>Approach to the situation of the problem in order to develop a plan to solve it.</p> <p>Internal task T2: State of the art</p> <p>Investigation of the precedents in that topic.</p> <p>Internal task T3: Project Proposal and Work Plan</p> <p>Creation of the needed documentation</p>	<p>Deliverables:</p> <p>Project Proposal and Work Plan</p>	<p>Dates:</p> <p>17/10/2017</p>

Table 1 – Work Package 1

Project: Set-up the Environment	WP ref: (WP2)	
Major constituent: Software and Hardware	Sheet 1 of 1	
<p>Short description:</p> <p>Set-up a computer with the necessary tools and software in order to be able to work in a programming environment of Keras.</p>	<p>Planned start date:13/10/2017</p> <p>Planned end date: 25/10/2017</p>	
	<p>Start event: State of the art</p> <p>End event: Test Keras on GPU</p>	
<p>Internal task T1: Install the necessary software</p> <p>Install in an Ubuntu OS the NVIDIA CUDA pack necessary for Tensorflow and then the Keras API.</p> <p>Internal task T2: Test Keras on GPU</p> <p>Make a test with an example of CNN in order to check if the computer is prepared for executing a CNN in Keras with GPU.</p>	Deliverables:	Dates:

Table 2 – Work package 2

Project: Prepare Database	WP ref: (WP3)	
Major constituent: Analysis and Software	Sheet 1 of 1	
<p>Short description:</p> <p>Acquire the data set from World Intellectual Property Organization (WIPO) in order to get as much data as we can, to train the system.</p>	<p>Planned start date:20/10/2017</p> <p>Planned end date: 17/11/2017</p>	
	<p>Start event: Test Keras on GPU</p>	

	End event: Resize images	
<p>Internal task T1: Select data</p> <p>Select the desired categories, in the Vienna Classification, we want to get in order to have the ones that it fits better in our system.</p> <p>Internal task T2: Prepare Dataset</p> <p>Create a Script to Download the logos from the Web page of WIPO by categories.</p> <p>Internal task T3: Resize images</p> <p>Reshape all images to the same size in order to feed properly the CNN.</p>	Deliverables:	Dates:

Table 3 – Work Package 3

Project: Evaluate Algorithms	WP ref: (WP4)	
Major constituent: Documentation	Sheet 1 of 1	
<p>Short description:</p> <p>With the information of the investigation about the state of the art, we have to evaluate which kind of CNN it's better for our case.</p>	<p>Planned start date:17/11/2017</p> <p>Planned end date: 25/12/2017</p>	
	<p>Start event: Resize images</p> <p>End event: Selection of CNN</p>	
<p>Internal task T1: How to feed the CNN?</p> <p>In order to know which is the best way to put the data in the CNN, we will try to do some experiments with different formats of data.</p> <p>Internal task T2: Selection of CNN</p> <p>For having best results we have to consider if it's better to create a new CNN and train with all of our data or take another CNN already created and take the final layers and train with our data.</p> <p>Internal task T3: Critical Review</p>	Deliverables:	Dates:
	Critical Review	18/12/2017

Table 4 – Work Package 4

Project: Improve Results	WP ref: (WP5)	
Major constituent: Analysis and Software	Sheet 1 of 1	
Short description:	Planned start date: 25/12/2017	

Tuning the CNN in order to accomplish the best results in terms of accurate classification.	Planned end date: 22/01/2018	
	Start event: Tuning End event: Extract statistics	
<p>Internal task T1: Tuning</p> <p>Create more layers in the CNN or change the models of the layers of the CNN.</p> <p>Internal task T2: Obtain the final Results</p> <p>When the tuning is done, we have to extract and prove the results.</p>	Deliverables:	Dates:

Table 5 – Work Package 5

Project: Present Results	WP ref: (WP6)	
Major constituent: Documentation	Sheet 1 of 1	
<p>Short description:</p> <p>Present the problem and the results related to it so a third person which hasn't been involved in the project can understand.</p>	Planned start date: 22/01/2018	
	Planned end date: 22/02/2018	
	Start event: Extract Statistics	
	End event: Finish Documentation	
<p>Internal task T1: Extract Statistics</p> <p>Make a visual way to show the final results with statistics, graphics...</p> <p>Internal task T2: Finish Documentation</p> <p>Complete all the documentation needed for the project.</p>	Deliverables: Final Report	Dates: 22/02/2018

Table 6 – Work Package 6

Milestones

WP#	Task#	Short title	Milestone / deliverable	Date (week)
1	1	Problem description		1-2
1	2	State of the art		2-3
1	3	Project Proposal and Work Plan	Project proposal and work plan	4
2	1	Install Software		3
2	2	Test Keras on GPU		4
3	1	Select Data		4
3	2	Prepare Dataset		4-6

3	3	Resize Images		7
4	1	How to feed the CNN		8
4	2	Selection of CNN		9-12
4	3	Critical Review	Critical Review (midterm)	11-12
5	1	Tuning		13-14
5	2	Obtain the final results		15-16
6	1	Extract Statistics		17-18
6	2	Finish Documentation	Final Review	17-21

Table 7 – Milestone list

1.3.3. Work plan modifications and Incidences

Incidents

From the beginning of the project the expected simplest work packages, especially setting up the development environment and preparing the train and test datasets, have been the most troublesome.

The available hardware resources were giving problems and since we haven't changed them, we couldn't set-up the environment properly. So we invested a lot of time in a task, which we have expected that would work out-of-the-box.

In case of the problem with the database, we aimed to extract the data from the web page of WIPO following the Vienna Classification. But then the WIPO dataset was not available as open data and we could only extract a small fraction of the image data of each class, far too small to train a convolutional neural network (CNN).

Thus, we searched for another source of visual trademark data and we found the *Oficina Española de Patentes y Marcas* (OEPM). We had to create a different way to extract the data with the use of bash script and python executables.

Finally, the process to obtain the results and extract statistics was longer because of the different methods we wanted to evaluate.

Work Plan modifications

As we can see in the comparison between the Project Proposal and the Final Report Gantt Diagrams below, the main structure it's quite the same. The most important modifications have been in the application of the execution and delivery of the tasks. Thus, the most important changes are in terms of execution time.

1.3.4. Time Plan (Gantt Diagram)

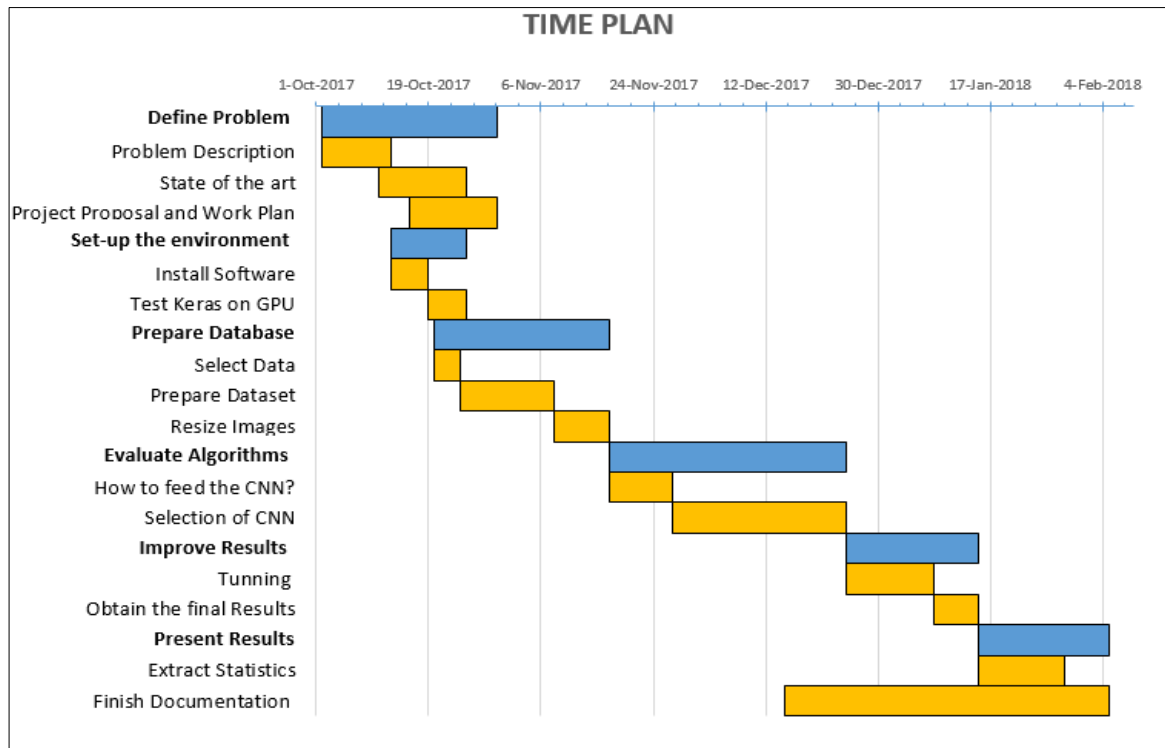


Figure 3 – Project Proposal Gantt Diagram

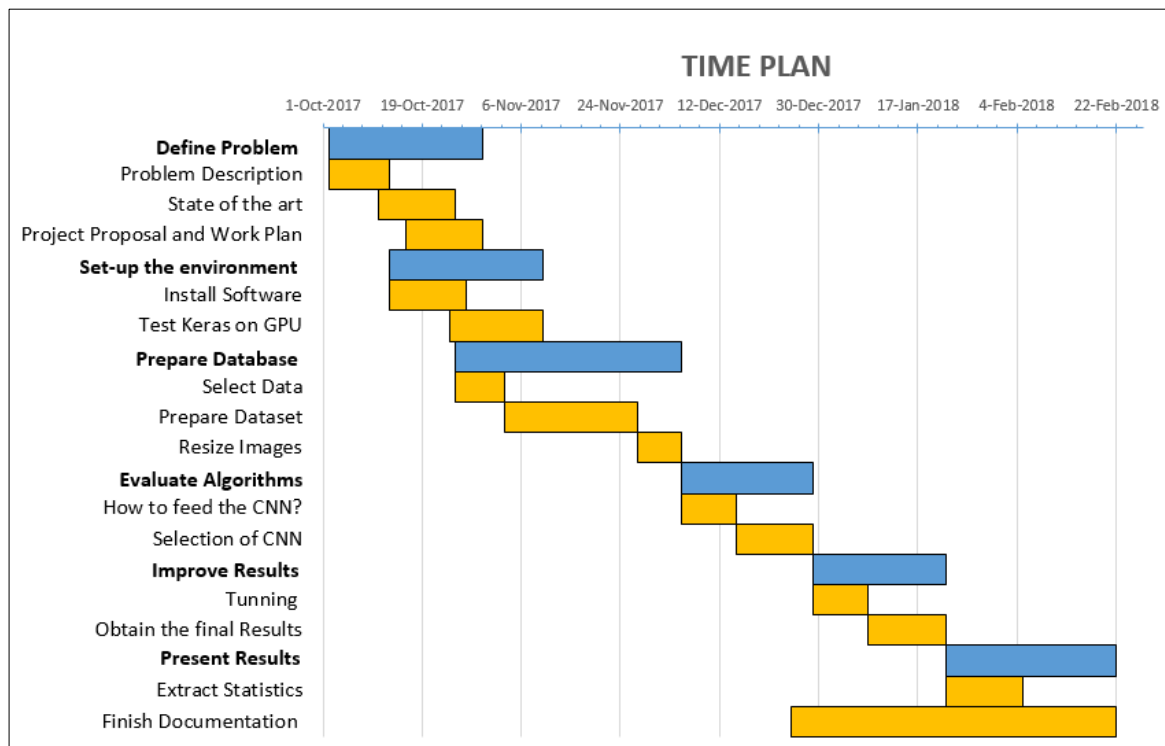


Figure 4 – Final Report Gantt Diagram

2. State of the art of the technology used or applied in this thesis:

Over the last years, deep learning methods have been shown to outperform previous state-of-the-art machine learning techniques in several fields, with computer vision being one of the most prominent cases. Thus, in order to understand the actual state of the art we have to take a look back to the beginning of deep learning and CNNs.

2.1. Evolution of CNNs

This technique has been applied to visual tasks since the 1980s. But despite few uses, the boom was in the mid-2000s. And why this fast expansion at this time? Basically this was the time when some technological advances appear. The developments in computing power, the coming on of a large amount of labelled data, the improvement of algorithms and the contribution of the research community to put on the forefront of the neural network renaissance, makes a really fast progression since 2012. But why is it so important, 2012?

Mainly because of the appearance of the one that started it all: AlexNet. The paper [1], has been cited a total of 19,882 times according to Google Scholar and it's one of the most influential publications in this field. The authors created a large deep CNN that was used to win the 2012 ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). This competition marked then the state of the art of CNNs and deep learning for computer vision in terms of image classification. Therefore, this event was so important that year because was the first time that a model performed so well on a historically difficult ImageNet dataset and the techniques of that experiment are still used nowadays, so they illustrated the benefits of CNNs in computer vision environment.

After this great event, all the new researches on that field have a significant relationship with AlexNet CNN model. In 2013 the winners of the ImageNet [2] competition based his architecture on the previous model, but they achieved very keys ideas about the behind of the scenes of the CNNs. So they tried to grasp how the inside part of the ConvNets works and they determined a really good way to visualize the filters and weights.

If we advance forward in time we can find in 2014 that another step was made. This time the paper [3] reinforced the notion that CNNs have to have a deep network of layers in order for this hierarchical representation of visual data to work. Thus, VGG16 Net created a new approximation to AlexNet with different size of filters and it's one of the most influential paper because of the idea to keep the model deep will keep it simple.

Following this thinking, Google with his paper [4] achieved a better score with the GoogLeNet model. But the point of this model is that introduces the idea of the Inception. This concept consists of a network inside a network layer. So it presents the non-sequential module which also leads to improve the performance and the computational efficiency.

One of the most exponents' models of deep CNN is the ResNet [5] architecture developed by Microsoft. In 2015 set a new record in classification, detection and localization in the number of layers of a network with 152. In this case, they won the ILSVRC with an error rate of 3.6%, which could represent a better score than humans with skills and expertise in this field.

What this model pointed out is the idea of residual learning. So they tried to reflect the idea that it's easier to optimize the residual mapping than to optimize the original mapping.

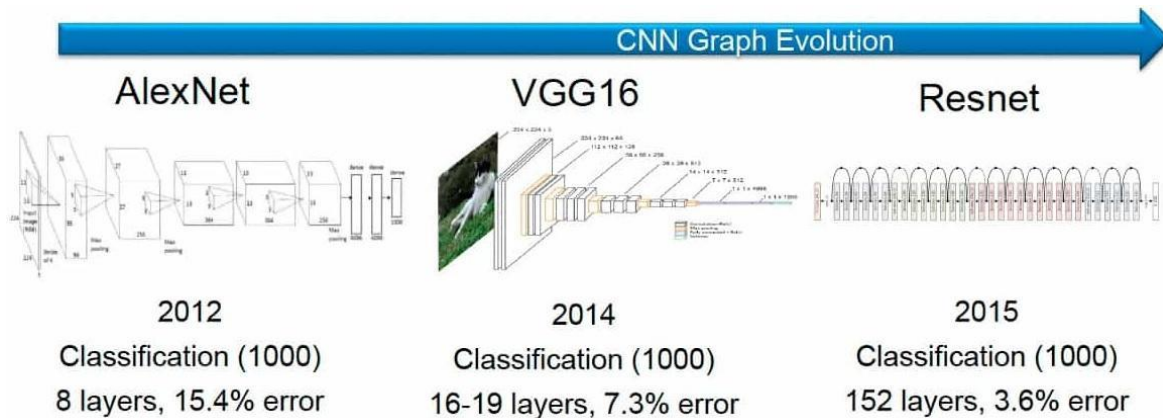


Figure 5 – CNNs Evolution

After this revolutionary idea of the residual CNNs, the state of the art is fluctuating around it and the most representative papers and works we have nowadays of deep ConvNets are based on that.

If we think about the last two mentioned models (GoogLeNet and ResNet) we can raise the question of whether there are any benefit in combining the Inception architecture with residual connections. So this is what they proof with the Inception-ResNet model in the paper [6].

Another question that could appear in our minds is why not going even deeper into the residual networks. This is what some researchers presented in the paper [7]. Thus, what they suggest is to apply a Residual Networks of Residual Networks (RoR).

2.2. From CPU to GPU

In order to understand the actual state of the art, we have to take a look at the developments in computing power.

Usually, the focus of the computer technology industry was centered in Central Processing Units (CPUs), because the general purpose of processors was for internet browsing, mobile apps, video streaming... which are based on integer operations and tend to be sequential. On the other hand, deep learning architectures are based on floats and decimal operations that typically are parallel.

Thus, deep learning demands a different type of processors with highest performance, which in this case is achieved with Graphics Processing Units (GPUs). Consequently by doing all the operations in parallel, so all at the same time, we can solve the usual problem of neural networks that need a huge amount of time to compute a simple model.

As we can see in the figure below, in the mid-2000s the CPU and the GPU computing power was not that different. At this time the GPUs were mostly used for gaming and for complex simulations, so they were mainly for graphics computations. But then the use for deep learning made increase the investments in the GPUs and the processing speed growth exponentially.

Nvidia's GPU Accelerates x86 CPUs Processing Speed

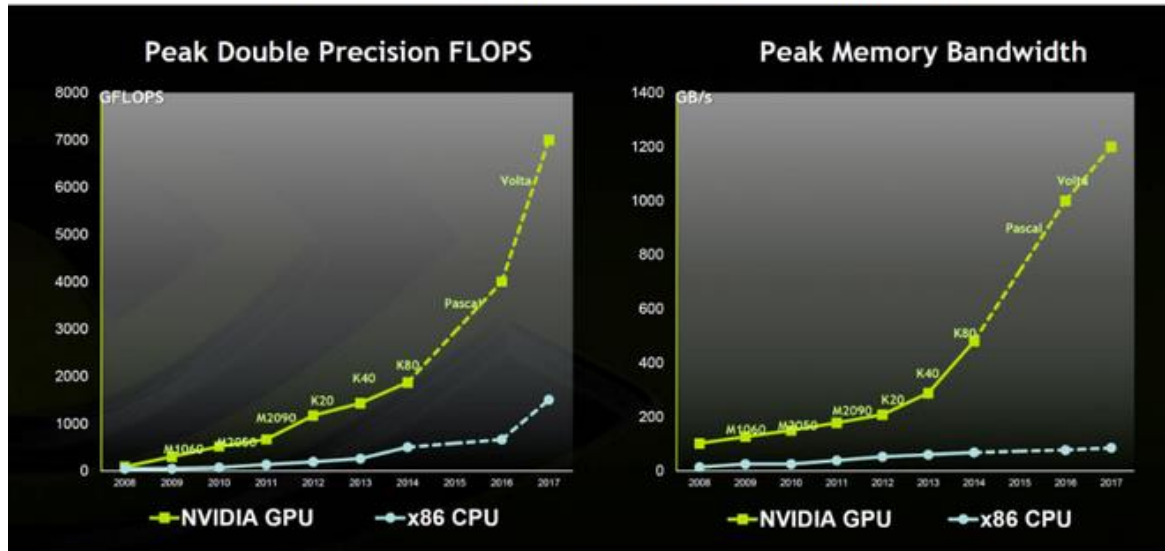


Figure 6 – CPU vs GPU processing speed [Source: [Nvidia's Presentation](#)]

Therefore, deep learning is unique enough to justify new architecture and support a really huge cost of ongoing chip development. Consequently, deep learning established a new state of the art in processing speed in terms of GPUs.

As we have seen previously, the state of the art is so difficult to define because every year and nearly every month new models appear and papers that overcome the last ones.

Therefore, the objective of the thesis was not overcoming the actual state of the art in the field. The goal was to be able to use and understand the techniques that are dominating the actual research. Thus, we combine the last researches in CNN models with the available GPU provided to prove that the actual state of the art could be achieved for everybody who has some few GPU resources and some knowledge about the CNNs and deep learning.

3. Methodology and project development:

This section aims to explain the methodology and the steps used to develop the project. We have explained before in the Work Plan segment the planned path to reach the final objective but in terms of time and resources. Now we are going to explain the details and the background of the methodology used in the project in order to construct a complete image of the process followed.

3.1. Set up the environment

In order to develop the thesis, we needed to set up the lab environment. The planned tool for developing the thesis has been, "Keras: The Python Deep Learning Library". But why use Keras?

As the home page of the Keras Documentation says: Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay, is key to doing good research.

Basically, we used Keras rather than any other because prioritizes the developer experience, the board adoption in the industry of deep learning and the research community, and among a lot of other reasons, the strong GPU support that makes it as fast as the other frameworks..

But before install Keras in our computer we needed to accomplish some previous requisites:

- First of all, following all the recommendations, we installed a Linux workstation. In our case we used Ubuntu 16.04 as our operating system for the computer.
- Secondly, we needed one backend engine. As [Keras](#) documentation recommends, we used [TensorFlow](#) backend.
- In order to run TensorFlow with GPU support, we must install the NVIDIA software. In our case we had available the GPU "Nvidia Geforce GTX 780". So we needed to install the CUDA toolkit 9.0, the associated drivers to this graphics card, and then the cuDNN v7.0. CUDA toolkit is a parallel computing platform and programming model that makes using a GPU for general purpose computing simple and fast. cuDNN is the NVIDIA CUDA Deep Neural Network library, which is a GPU-accelerated library of primitives for deep neural networks.
- For installing TensorFlow, we did it through the Virtualenv installation. Virtualenv is a virtual Python environment isolated from other Python environments, incapable of interfering with or being affected by other Python programs on the same machine.
- Then, when we had installed TensorFlow, with the version of Python, in our case Python 3.6, we could install Keras: The Python Deep Learning library.

3.2. Database

In order to perform this thesis, one of the most important things are the database. So we needed to find a well classified and tagged database for then make the training easier. And in terms of trademark databases and intellectual property we had recourse to WIPO.

3.2.1. WIPO and Vienna Classification

World Intellectual Property Organization (WIPO) is the global site for intellectual property services, policy, information and cooperation. It is an agency of the United Nations created in 1967, which mission is to lead the development of a balanced and effective international intellectual property system that enables innovation and creativity for the benefit of all. This organization is the responsible of the Vienna Classification.

Vienna Classification (VCL), established by the Vienna Agreement, is an international classification system used to classify the figurative elements of marks. The Vienna Agreement is the WIPO-administered multilateral treaty that establishes the Vienna Classification that was signed on June 12, 1973.

Thus, we used the Vienna Classification because has the advantage that contains figurative elements categorised according to a single classification system. This classification is applied to 60 cataloguing organizations around the world. Also, it is continuously revised and every five years a new edition is published. In our case, we used 7th edition of the Classification, which comprises from January 1, 2013 to December 31, 2017.

3.2.1.1. Structure of Vienna Classification

The classification scheme constitutes a hierarchical system that proceeds from the general to the particular, dividing all figurative elements into categories, divisions and sections. This publication that we used contains a total of:

- 29 categories,
- 145 divisions,
- 806 main sections, and
- 903 auxiliary sections.

Every figurative element in a section is referred by three numbers: the first, which denotes the category, may be any number from 1 to 29; the second, which denotes the division, may be any number from 1 to 19; and the third, which denotes the section, may be any number from 1 to 30.

Category	Name of Category
01	CELESTIAL BODIES, NATURAL PHENOMENA, GEOGRAPHICAL MAPS
02	HUMAN BEINGS
03	ANIMALS
04	SUPERNATURAL, FABULOUS, FANTASTIC OR UNIDENTIFIABLE BEINGS
05	PLANTS
06	LANDSCAPES
07	CONSTRUCTIONS, STRUCTURES FOR ADVERTISEMENTS, GATES OR BARRIERS
08	FOODSTUFFS
09	TEXTILES, CLOTHING, SEWING ACCESSORIES, HEADWEAR, FOOTWEAR
10	TOBACCO, SMOKERS' REQUISITES, MATCHES, TRAVEL GOODS, FANS, TOILET ARTICLES
11	HOUSEHOLD UTENSILS
12	FURNITURE, SANITARY INSTALLATIONS

13	LIGHTING, WIRELESS VALVES, HEATING, COOKING OR REFRIGERATING EQUIPMENT, WASHING MACHINES, DRYING EQUIPMENT
14	IRONMONGERY, TOOLS, LADDERS
15	MACHINERY, MOTORS, ENGINES
16	TELECOMMUNICATIONS, SOUND RECORDING OR REPRODUCTION, COMPUTERS, PHOTOGRAPHY, CINEMATOGRAPHY, OPTICS
17	HOROLOGICAL INSTRUMENTS, JEWELRY, WEIGHTS AND MEASURES
18	TRANSPORT, EQUIPMENT FOR ANIMALS
19	CONTAINERS AND PACKING, REPRESENTATIONS OF MISCELLANEOUS PRODUCTS
20	WRITING, DRAWING OR PAINTING MATERIALS, OFFICE REQUISITES, STATIONERY AND BOOKSELLERS' GOODS
21	GAMES, TOYS, SPORTING ARTICLES, ROUNDABOUTS
22	MUSICAL INSTRUMENTS AND THEIR ACCESSORIES, MUSIC ACCESSORIES, BELLS, PICTURES, SCULPTURES
23	ARMS, AMMUNITION, ARMOUR
24	HERALDRY, COINS, EMBLEMS, SYMBOLS
25	ORNAMENTAL MOTIFS, SURFACES OR BACKGROUNDS WITH ORNAMENTS
26	GEOMETRICAL FIGURES AND SOLIDS
27	FORMS OF WRITING, NUMERALS
28	INSCRIPTIONS IN VARIOUS CHARACTERS
29	COLOURS

Table 8 – Vienna Classification Categories

3.2.2. Data extraction

After trying to extract the data directly from the WIPO database, and not succeeding because it wasn't open data, we found the Oficina Española de Patentes y Marcas (OEPM). This office has the OPENDATA project, which consists on publish public data in order to be reusable for third parts for using it to their particular applications. This data is open and available at the following link:

<https://sede.oepm.gob.es/eSede/datos/es/catalogo/datos.html?catalogo=marcas#tabs-biblio>

This database is the collection of daily generated data. Thus, you can find the files generated in one concrete day, one month or all year. These files are stored in compressed folders that inside them we have the division of the different months and days of the year, which the data is stored in XML format.

All the XML files of each entry have the parameter of which category belongs this trademark. Therefore, what we did was change the way that the database was organized, from sort by date to sort by category. In order to do that, we extracted from each XML the parameter of the associated category, and we stored this XML files sorted by categories. Once we had all the XML files of every category, we needed to extract from the XML the link to download every image. These images links corresponded to the database of the European Trade Mark and Design Network ([EuropeanTMDN](#)). The problem with that organization was that the connections to download images were limited to 1 connection every 2 hours. So then the process to download the images of every category was much more slowly than we expected.

Finally, after combining some tools of Ubuntu and Python we extracted around 150 thousand images from 29 different categories, belonging to the years from 2015, 2016 and 2017.

3.2.3. Working Datasets

As we commented in the last section, the process to download the images was quite slowly. Thus, we used the data as soon as we had.

Once we extracted all the images from all the categories of 2015 folder, we realized that the amount of data available was not uniform in the categories as the range of images from one to another was so different. We had some categories with really few data around 60 images and then some others with around 20 thousand images. The main issue was that we had to choose some classes with quite generic images in order to have enough data, but not so generic that every image can belong to this category. For example, the categories 26, 27 and 29, as we can see in the Vienna Classification table, they are so general that we don't have a strong feature that can define this class.

So finally, we chose the 5 classes with highest amount of data and with the strongest feature, which makes easily to differentiate from one class to another. These classes were:

- 01 CELESTIAL BODIES, NATURAL PHENOMENA, GEOGRAPHICAL MAPS
- 02 HUMAN BEINGS
- 03 ANIMALS
- 05 PLANTS
- 24 HERALDRY, COINS, EMBLEMS, SYMBOLS

Thus, in the development of the project we were able to obtain two different datasets while we were downloading the data.

- **First dataset:** with the 5 classes already mentioned before and an average of 2000 images per each category belonging to 2015 database. In this dataset there wasn't applied any type of image processing technique in order to improve the images. So we had a dataset divided in 1000 images of Train, 500 images of Validation, and 375 images of Test.
- **Second dataset:** also with the 5 classes cited but in this case belonging to 2015 and 2016 database. The average of images per category is around 6500, and in this occasion we deleted all the duplicated images inside the same category. In order to split the dataset into three non-overlapping sets we called Train, Validation and Test, we picked the images randomly trying to avoid any sorting pattern, as sorted by name or date as the computers usually try to sort the files. So this dataset was divided in 4032 images of Train, 1008 in Validation and 1260 in Test.

This obtained datasets were splitted following the concept of cross-validation: 80% of the data was destined for Train and 20% for Test. After that the training set was splitted in 80% train and 20% validation.

3.3. CNNs, the definitive algorithm

As we have seen in the section 2 [State of the art](#), CNNs are the state of the art in terms of solving image classification tasks. Thus, there isn't any doubt that CNNs are the definitive algorithm for our project.

A Convolutional Neural Network is a type of artificial neural network where neurons correspond to receptive camps in a very similar way to neurons in the primary visual cortex of a biological brain. Each neuron receives some inputs, performs a dot product and usually follows it with a non-linearity, which have some learnable weights and biases.

3.3.1. Overview of a CNN

Reviewing how a neural network works, it receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the "output layer" and in classification settings it represents the class scores.

3.3.1.1. Convolutional Layer

But the densely-connected layers of the neural networks don't work well with images because there is a huge number of parameters. Because of that we use convolution layers. The main difference between them is that dense layers learn global patterns in their input feature, while convolutional layers learn local patterns. We can see an example in the figure below.

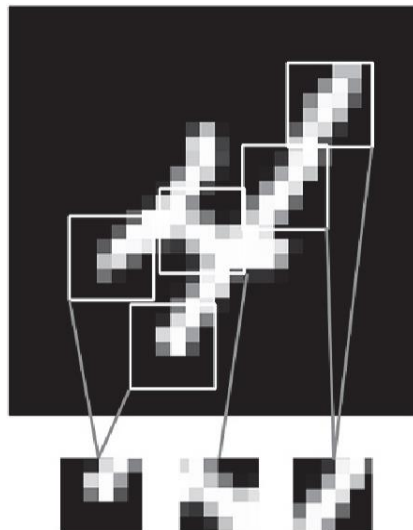


Figure 7- Local patterns division [Source: Deep Learning with Python]

The good thing about that is that the convolutional layers can recognize a certain pattern anywhere in a picture if they learned before this pattern, so they have the property to be translation-invariant. As they can learn spatial hierarchies of patterns, they can learn local patterns such as edges, but then in a second convolution layer they can learn related patterns to the first one.

ConvNets take advantage of the fact that the input consists of images. So, they operate over 3D tensors called "feature maps" with 3 different dimensions: width, height and depth. For example we worked generally with colour images of 200x200 pixels. This means that the input

feature map will be (200, 200, 3). But after, the convolution operation extracts patches from the input feature map, and apply a transformation to all the patches producing an output feature map. This output feature map has the same three dimensions but changing the depth depending on the parameter that have been given to the layer. This parameter of depth we could call "filter" and his function is to encode specific aspects of the input data, for example like one of these filters can be to encode the edges of a picture.

Thus, convolutions operations are defined by two parameters:

- The size of the structural element that you want to extract from the inputs. Typically it is 3x3 or sometimes 5x5.
- The depth of the output feature map, in other words, the amount of filters that you want to apply. Usually it is a binary base number, like 32 or sometimes 64.

3.3.1.2. Activation Function

The activation function is an abstraction that represents the rate of action potential firing in the cell. In other words, this is a binary function that gives the rate if a neuron is activated or not.

Basically in this project we used 2 different activation functions:

- **Sigmoid function:** this function has been used mainly in machine learning, especially for the logistic regressions and some basic neural networks.

The good point of that function is that the derivate is easy to calculate and save times for building models.

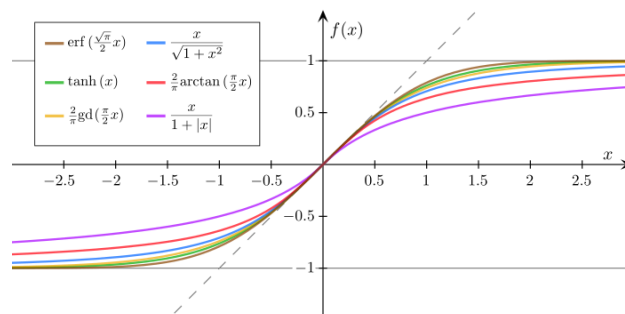


Figure 8 – Some sigmoid functions compared [Source: [wikipedia.org](https://en.wikipedia.org/wiki/Sigmoid_function)]

- **ReLU function:** nowadays, instead of sigmoids, deep learning networks use Rectified Linear Units (ReLUs) for the hidden layers. This function is defined as the positive part of its argument. $f(u) = u^+ = \max(0, u)$

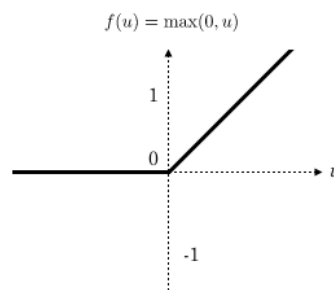


Figure 9 – ReLU function [Source: [researchgate.net](https://www.researchgate.net/publication/312211111)]

ReLU activations are the simplest non-linear activation function we can use, and the researchers has shown that is much faster for large networks.

3.3.1.3. Max Pooling Layer

In order to progressively reduce the amount of parameters and computation in the network we periodically insert a Pooling layer between successive Convolutional layers.

Max pooling basically consists in extracting windows (typically 2x2) from the input feature map and giving in the output the MAX value of each channel. In this way we achieve a downsampling of the feature map by a factor of 2.

So the main reason to use downsampling is to reduce the number of feature map coefficients to process, as well as to create a spatial filter hierarchies making that the convolutional layers look like an increasing large window.

Thus, after reviewing this concepts we can extract that CNNs commonly have the architecture shown in the figure below: Convolutional Layer + ReLU activation + MaxPooling + Fully Connected.

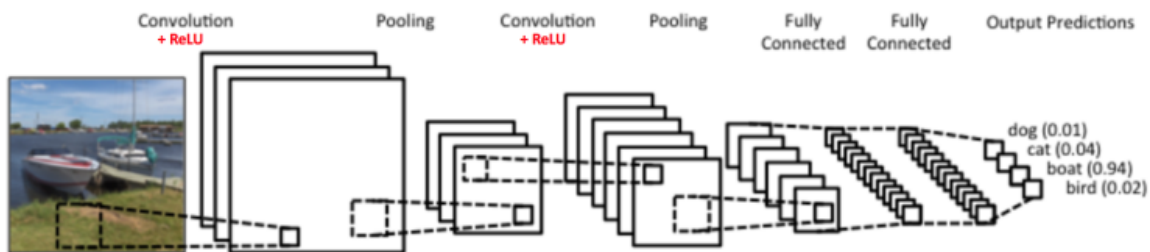


Figure 10 – Architecture of a typical CNN [Source: ujjwalkarn.me]

3.3.2. ConvNet from scratch

After seeing the state of the art of the technology we can think and admit that the first step to do is apply this already created and trained networks in our classification task. So then, why try to train and develop a CNN from scratch?

The main goal is to get the hands on the CNNs and understand how they work more clearly and uncover the black box that represents a CNN.

So, following the thinking that you will learn better and become more familiar with a topic doing the experiment and not just studying the theory, we wanted to create this new CNN.

The results and the way which we created the CNN will be explained in the section 4.

3.3.3. Pre-trained CNN

After creating the CNN from scratch and understanding how really works, we wanted a highly effective approach using a pre-trained ConvNet. A pre-trained network is an already saved network previously trained in a large dataset like a large-scale image classification task. In this case the large-scale dataset is ImageNet. So, what we wanted to do is to apply the generic model of the pre-trained ConvNet to our trademark image classification task. Thus, the idea was that as the logos are an abstracted concept from the original object, use this relation in order to match the real concept image trained in the pre-trained ConvNet on ImageNet, with the abstract concept of the trademark image. As we can see below in the comparison figure:



Figure 11 – Left side a real image of a dog[Source]; Right side the abstract logo based on the image[Source]

3.3.3.1. VGG Net

The first pre-trained network that we wanted to try was the VGG Net, because even if it is not the actual state of the art, it is a similar architecture to the ones we use in the scratch CNN and it is still a really optimum model.

The VGG network architecture was introduced by Karen Simonyan and Andrew Zisserman in their 2014 paper, “Very Deep Convolutional Networks for Large-Scale Image Recognition”.

This network is characterized by simplicity and depth. It uses only 3x3 convolutional layers, which comparing to the 11x11 of *AlexNet* and the 7x7 of *ZF Net* is a really big step. The authors give the reason to the fact that having 2 consecutive 3x3 filters give an effective receptive of 5x5, and also a 3 consecutive filters 3x3 give a receptive of 7x7 filters. This in turn simulates a larger filter while keeping the benefits of smaller filter sizes. One of the main benefits is that the number of parameters decrease considerably.

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
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 12 – Architectures of VGG Net; D produces the best results[Source: kdnuggets.com]

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

Figure 13 – Number of parameters of the configurations (in millions)

As we can see in the architecture of the VGG Net, we have 2 and 3 consecutive 3x3 filters using the ReLU layers activation after each convolution, alternated with a Max Pooling of 2x2. So, we can appreciate the simplicity of this network because we basically use Convolutional Layer, the ReLU activation, a max pooling to downsampling and fully connected layers to synthesize the classification.

3.3.3.2. Inception-ResNet

In order to make a big step forward in the networks used before, we wanted to proof our image classification task with the actual state of the art. So, as we commented in the section 2, Inception-ResNet is one of the most advanced CNN architecture.

This deep convolutional neural network is based in two different approaches performed in the recent years. One performance is the Inception (Inceptionv3, which is an updating of GoogLeNet) architecture that has achieved very good performances in terms of accuracy and loss with a relatively low computational cost. On the other hand, the most recent researches in the topic have introduced the term of residual connections (ResNet). This combination achieved the result shown in the paper: "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning".

The idea of **Inception**, is to make a process, which was typically sequential, works in parallel. Thus, when at each layer of a traditional ConvNet you have to make a choice of whether to have a pooling operation or a conv operation, an Inception module allows you to do is perform all of these operations in parallel. As shown in the next figures:

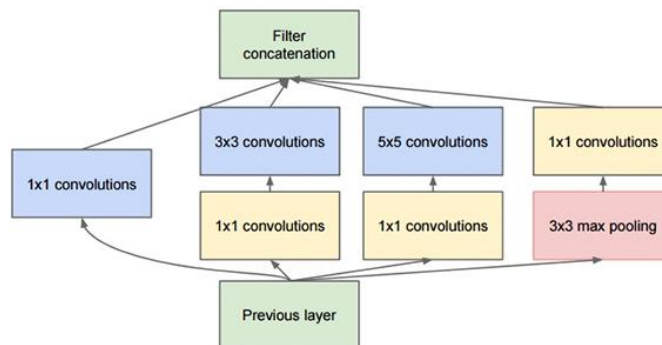


Figure 14 – Basic Inception module [Source: [Going deeper with convolutions](#)]

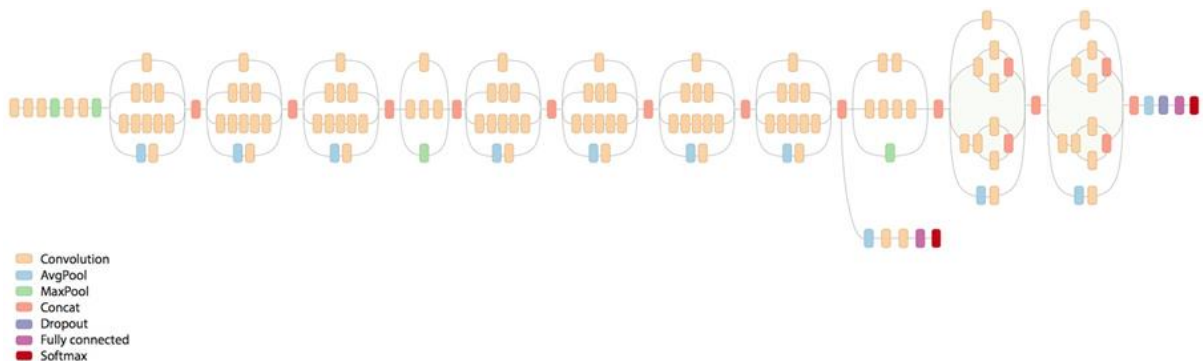


Figure 15 – GoogLeNet's architecture [Source: [hacktildawn.com](#)]

The idea of **residual**, is to add the original input in the result of the convolutional block. So, if the normal convolutional block would give you an $F(x)$, we will add the original input x and we would get $H(x) = F(x) + x$.

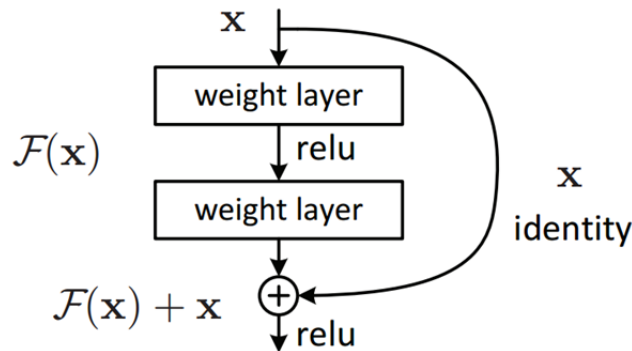


Figure 16 – Residual learning block [Source: [Deep Residual Learning for Image Recognition](#)]

Therefore, combining this two concepts we get the architecture of the Inception-ResNet Convolutional Neural Network. In the figure below we can see first the full network expanded, while in the second we have a compressed view where we can observe the structure of Inception with the parallel convolutional layers and also the Residual part adding in the result of the block.

Inception Resnet V2 Network

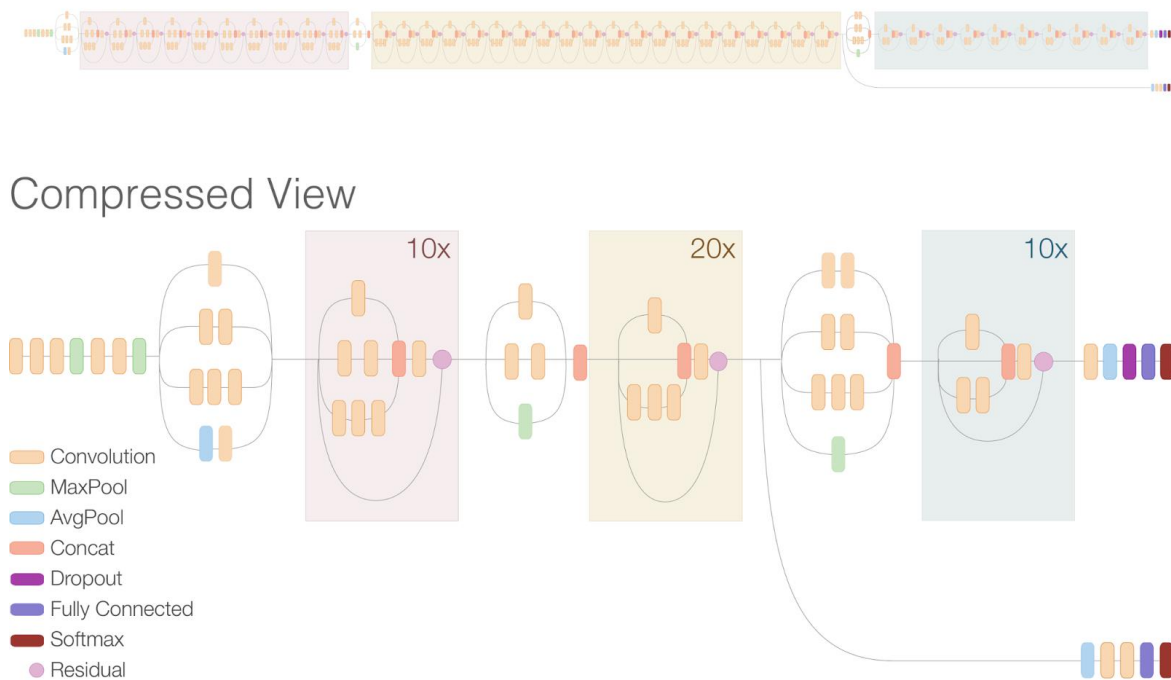


Figure 17 – Schematic diagram of Inception-ResNet [Source: [research.googleblog.com](#)]

3.3.4. Training and Evaluation Process

The process of training a deep learning architecture is similar to the Learning Process of humans. When a child encounter a new object never seen before, he will not know what it is. But then an adult point out that this object has a name and it is that, the child has to associate the image of that object with the name or label. In order to do that, the kid has to see so many times the same object in different environments or different shapes to be able to recognize when this object is truly.

Neural networks follow a similar process in order to train and then recognize the object or image with the correct label. And for improve the performance of the networks doing this difficult task we need some different techniques. There is some methods to do that but in our case we used the RMSprop.

The **RMSprop** (Root Mean Square Propagation) is a method in which the learning rate is adapted for each of the parameters. It divide the learning rate for a weight by running average of the magnitudes of recent gradients for that weight. So what it essentially do is normalize the gradient by dividing it by the magnitude of recent gradients. Consequently, when a plateau in the error surface is encountered and the gradient is very small, the updates take grater steps making a faster learning.

$$v(w, t) = \underbrace{\gamma}_{\text{MeanSquare}} \cdot v(w, t - 1) + (1 - \gamma) \cdot \underbrace{(\nabla Q_i(w))^2}_{\text{Square gradient of each weight}}$$

Forgetting factor (typically 0.9)

Dividing the squared gradient we obtain:

$$\text{RMSprop} \rightarrow w = w - \frac{\underbrace{\eta}_{\text{Learning rate (typically 0.001)}}}{\sqrt{v(w, t)}} \underbrace{\nabla Q_i(w)}_{\text{Gradient of weight}}$$

When we train the model we want to adjust the model's parameters in order to get our predictions closer and closer to the ground truth. Thus, we want to quantify the quality of a set based on how well the set agree with the ground truth labels.

For this task we need to calculate the score or loss function every time when we compute gradients for the weights in the network. To do this calculation we use the cross entropy loss function.

The **cross-entropy** function in our case is defined as a loss function for optimizing the weights of the network. What we want to do is to compare the true label and the predicted value of the current model.

So supposing that we are trying to train a neuron with input variables, x_1, x_2, \dots , corresponding weights w_1, w_2, \dots , and a bias b :

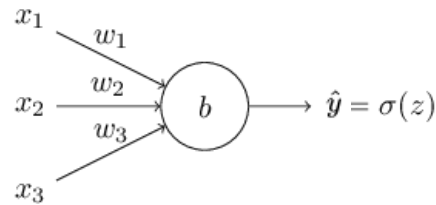


Figure 18 – Neuron model [Source: tensorfly.cn]

The output from the neuron is, $\hat{y} = \sigma(z)$ where we can define the cross-entropy cost function for this neuron by:

$$H(y, \hat{y}) = - \sum y \log \hat{y} = - \frac{1}{n} \sum_x y \log \hat{y} + (1 - y) \cdot \log(1 - \hat{y})$$

where n is the total number of items of training data, the sum is over all training inputs, x , and y is the corresponding desired output.

In summary, in order to do a well training and go forward in terms of result, we have used as a measure of loss the cross-entropy function and as a measure of accuracy the RMSprop function. Thus, the main function of these measures are to find the weights that minimize the loss function.

4. Results

For presenting the results we will follow the path we used to achieve the final result, passing to every remarkable test we did. Because of that we will expose the results following the different test did with every single model.

4.1. ConvNet from scratch

The main goal of the first tests were, as we said in another section, get the hands on the CNNs and understand how they work more clearly and undercover the black box that represents a CNN. So, we used the CNN created from scratch, totally new and with the few data that we had in that moment.

4.1.1. Simplest first model (1)

The simplest step, and with the few data we had, was to create a new CNN from scratch. We used the **First Dataset** mentioned before in section [Working Datasets](#). The architecture of the CNN designed is the following:

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d_1 (MaxPooling2)	(None, 99, 99, 32)	0
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18496
max_pooling2d_2 (MaxPooling2)	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 46, 46, 128)	73856
max_pooling2d_3 (MaxPooling2)	(None, 23, 23, 128)	0
conv2d_4 (Conv2D)	(None, 21, 21, 128)	147584
max_pooling2d_4 (MaxPooling2)	(None, 10, 10, 128)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_5 (MaxPooling2)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
dense_2 (Dense)	(None, 5)	2565

Figure 19 – Architecture of the first model

As we can observe we have a combination of convolutional layers with ReLU activation and Max Pooling to downsample the parameters of the network. We applied an increasing number of filters in every convolution, starting from 32 and ending with 128. Then with the Flatten layer we convert the output of the last convolution applied into a 1D feature vector in order to create a single long feature vector to be used by the dense layer for the final classification. If we take a look at the *max_pooling2d_5*, we can see a vector of (4, 4, 128) and if we multiply this numbers and we put in a single vector we get the (2048) size vector. Finally, we need to use dense layers in order to make the classification. Since we are attacking a multi-class classification problem, we are ending the network with a Dense layer of the size of the classes that we have, in our case 5.

With this model we could achieve an accuracy of 80% in the Train and Validation and in terms of loss we could achieve around 50%. As we can see in the figures below we trained this model during 5 epochs with 20 steps per epoch.

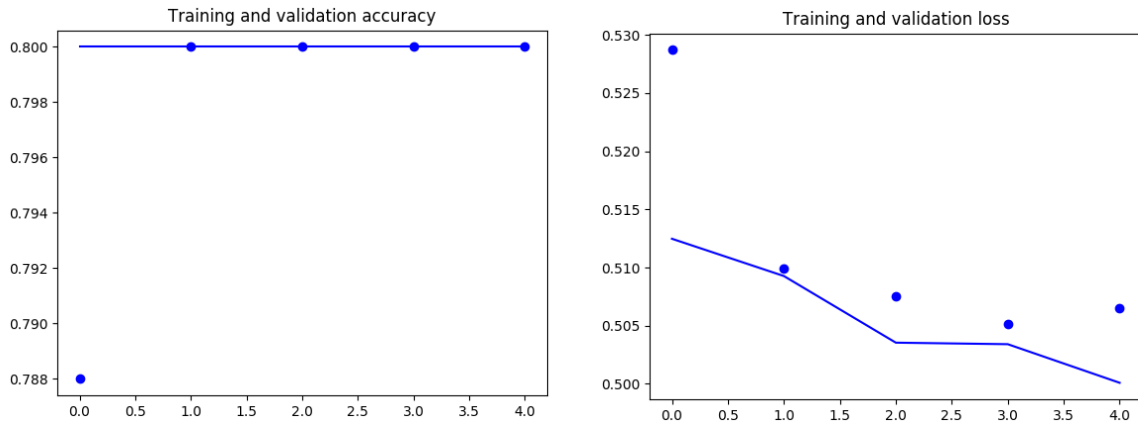


Figure 20 – Accuracy and loss of the first model

Note: Training values are dots and validation values are solid lines.

4.1.2. Update of the first model (2)

In order to overcome some problems of the first test like overfitting, we decided to add data augmentation to our model. Overfitting is caused by having too few samples to learn from, and in our case we had only 1000 images to train, 500 for validating and 375 for test. The function of data augmentation is to generate more training data from already existing samples via random transformation to these images. So with this process at training time the model would never see the exact same picture twice.

Layer (type)	Output Shape	Param #
conv2d_1 (Conv2D)	(None, 198, 198, 32)	896
max_pooling2d_1 (MaxPooling2D)	(None, 99, 99, 32)	0
conv2d_2 (Conv2D)	(None, 97, 97, 64)	18496
max_pooling2d_2 (MaxPooling2D)	(None, 48, 48, 64)	0
conv2d_3 (Conv2D)	(None, 46, 46, 128)	73856
max_pooling2d_3 (MaxPooling2D)	(None, 23, 23, 128)	0
conv2d_4 (Conv2D)	(None, 21, 21, 128)	147584
max_pooling2d_4 (MaxPooling2D)	(None, 10, 10, 128)	0
conv2d_5 (Conv2D)	(None, 8, 8, 128)	147584
max_pooling2d_5 (MaxPooling2D)	(None, 4, 4, 128)	0
flatten_1 (Flatten)	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 512)	1049088
dense_2 (Dense)	(None, 5)	2565
Total params: 1,440,069		
Trainable params: 1,440,069		
Non-trainable params: 0		

Figure 21 – Architecture of the update of the first model

If we take a look at the architecture of the model, we can appreciate just one difference between the first model and the update, we added a dropout layer to further fight overfitting. The dropout layer consists on during training, half of neurons on a particular layer will be deactivated. This dropout is used in the fully connected layers.

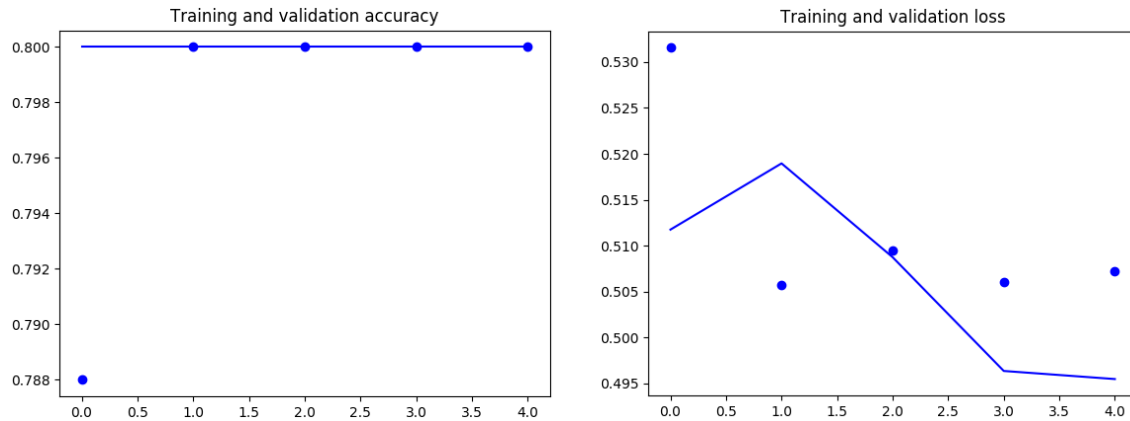


Figure 22 – Accuracy and loss of the update of the first model

Analysing the plots, we can see that even if we added data augmentation and the dropout we didn't achieve a big change. The only difference is that we reduced to 49.5% the loss.

4.2. Pre-trained VGG16Net

As we had a small image dataset, we couldn't perform a really optimum model. Because of that we used a pre-trained network, which gave us a highest effective approach. So what we did was profit the fact that this VGG16Net was pre-trained on ImageNet, where there is more than 1.4 million images and around 1000 different categories. As we couldn't compete with this type of models, we chose the option to use it.

4.2.1. VGG16Net with First Dataset (3)

The first test that we did with a pre-trained network was with few data, which means the first dataset. We already revised the architecture of the VGG16Net in the section [VGG Net](#), and was composed of convolutional layers with ReLU activation alternating with Max Pooling layers. This is really similar as the architecture we designed in the CNN from scratch, but with some more convolutions concatenated and of course trained with more data. When we are loading the VGG16 model we have to include some arguments. In order to specify the weights of the model we can initialize the weights of ImageNet and not the random weights that will initialize a normal network.

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 6, 6, 512)	14714688
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 512)	9437696
dense_2 (Dense)	(None, 5)	2565
Total params: 24,154,949		
Trainable params: 24,154,949		
Non-trainable params: 0		

Figure 23 – Architecture of the VGG16 model adapted to our task

If we observe the figure above, we can see that what we did is just add the last layers of the network in order to adapt the step of classification to our task. But before compiling and training the model we had to freeze the convolutional base. This was made for preventing that the weights pre-trained before in the model get updated during the new training process. If we didn't apply technique, the representations that were previously learned by the convolutional base would get modified.

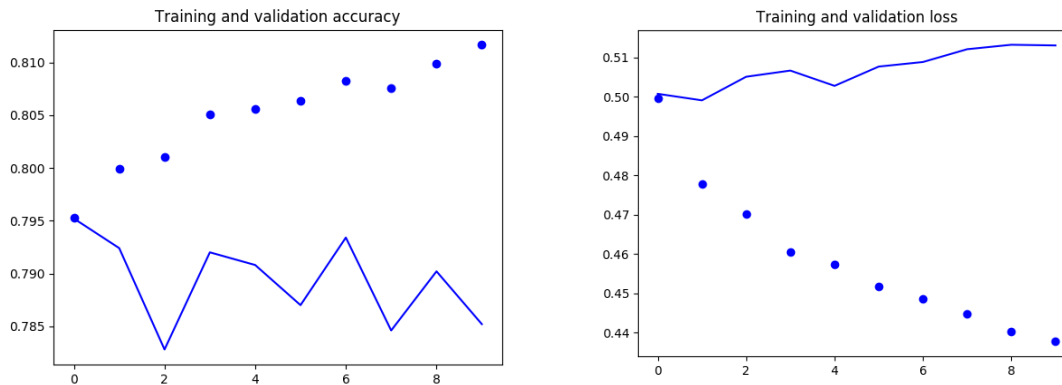


Figure 24 – Accuracy and loss of the VGG16 with First Dataset

Taking a look on the figure 24, we can see that the maximum accuracy in training is a bit higher than 81%, meanwhile in validation the higher accuracy we achieved is in the first epoch and then we go down till 77%. If we observe the loss plot, we extract the same behaviour; the training loss decrease even less than 44% but the validation goes from 50% to close than 52% in the last epochs. This result makes us conclude that we have a strong overfitting. This means that the model has adapted really well to the training data but then the validation data, as it is different, the model don't success in the same way.

4.2.2. VGG16 applying fine tuning (4)

Observing the last tests we can appreciate that we have adapted so much the model to the database causing some overfitting. So this mainly was caused by the problem of having so few data, and in this model finally we could increase the data using the **Second Dataset**. In this case, as we mentioned in the section [Working Datasets](#), we had 4032 images for training, 1008 for validation and 1260 for test.

In order to find in a better way the main features of the database and extract global structures that can avoid this overfitting we used the technique of fine tuning. It consists on unfreeze a few of the top layers of a network, and training them in a new way jointly to the added fully connected layers. So what we pretend is to slightly adjust the more abstract representations of the model to make it more relevant for the task we are doing.

Thus, if we have in mind how is the VGG16 Net, in the next figure we can observe the 5th block of successive convolutional layers and the end layers of the network. What we did was unfreeze all the layers till the *block4_pool*, that means unfreeze the layers *block5_conv1*, *block5_conv2* and the *block5_conv3*, which are the 3 last convolutional layers.

block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Convolution2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Convolution2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Convolution2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
flatten (Flatten)	(None, 25088)	0
fc1 (Dense)	(None, 4096)	102764544
fc2 (Dense)	(None, 4096)	16781312
predictions (Dense)	(None, 1000)	4097000

Figure 25 – Last layers of VGG16 Net

So doing this operation we will have the model of the following figure:

This is the number of trainable weights before freezing the conv base: 30
 This is the number of trainable weights after freezing the conv base: 10

Layer (type)	Output Shape	Param #
vgg16 (Model)	(None, 6, 6, 512)	14714688
flatten_1 (Flatten)	(None, 18432)	0
dense_1 (Dense)	(None, 512)	9437696
dense_2 (Dense)	(None, 5)	2565

Total params: 24,154,949
 Trainable params: 16,519,685
 Non-trainable params: 7,635,264

Figure 26 – VGG model applied fine tuning

As we can see, we have the model of the VGG16 Net and then the added layers in order to make the classification part. But looking more carefully, we can see that the number of trainable weights before freezing all the convolutional base would be 30, so all the layers of the model. Nevertheless when we unfreeze some layers of the last convolutional block, or in another way we freeze all the convolutional base except the block 5 of convolutional layers, we have a number of trainable weights of 10. And if we take a look on the trainable parameters, we can observe that we reduce to a great extent the number of trainable parameters.

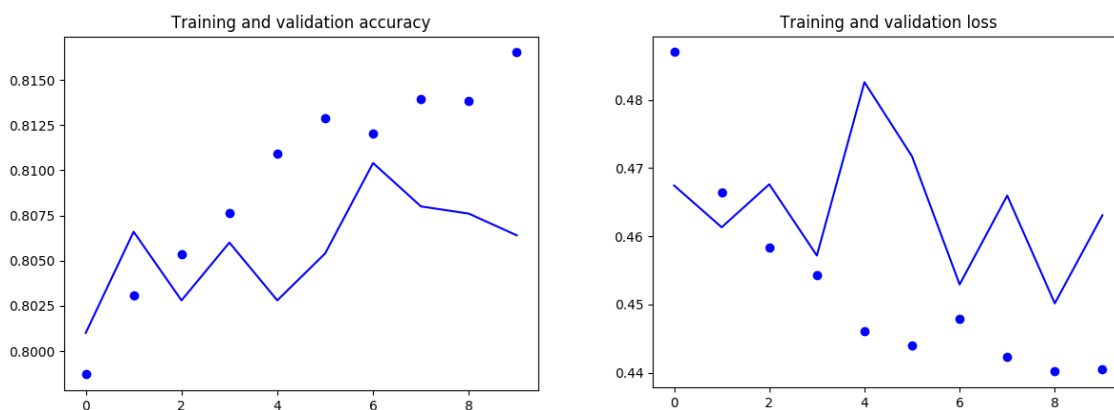


Figure 27 – Accuracy and loss VGG16 applied fine tuning

Observing the accuracy and loss of the model, we can see the values of training from 80% to 82% approximately, and the validation they oscillate from 80% to 81% but in an ascending projection. Looking to the loss rate, the training goes from 49% to less than 44%, and the validation also oscillate from 47% to 45% but in a decreasing tendency.

Analysing this plots we can figure out that we achieved a better model in terms of overfitting because of the tendency of the training and validation is not opposite. So even the results are quite similar to the other models in terms of accuracy or loss, we can say that in general terms that we have a most complete and adapted model. There's also a part that helped in order to improve the overfitting and is the fact that we had more data thanks to the second dataset.

4.3. Inception-ResNet (5)

After all the past test and models we wanted to make an step forward with the project using one of the most advanced tools in the state of the art, Inception-ResNet CNN.

As we have seen in the section [Inception-ResNet](#), the network have a really complicated architecture, so we can't train again all the network in our model. Therefore what we did is the same as the last model, do a fine tuning. In this case we can observe in the figure below the last layers of the convent. We have a combination of convolutions and other functions. So what we did was unfreeze the layers till the *block8_10_mixed*, having to train in this way the convolution of the *block8_10_conv*, and the *conv_7b*.

block8_10_mixed (Concatenate)	(None, 4, 4, 448)	0	activation_200[0][0] activation_203[0][0]
block8_10_conv (Conv2D)	(None, 4, 4, 2080)	933920	block8_10_mixed[0][0]
block8_10 (Lambda)	(None, 4, 4, 2080)	0	block8_9_ac[0][0] block8_10_conv[0][0]
conv_7b (Conv2D)	(None, 4, 4, 1536)	3194880	block8_10[0][0]
conv_7b_bn (BatchNormalization)	(None, 4, 4, 1536)	4608	conv_7b[0][0]
conv_7b_ac (Activation)	(None, 4, 4, 1536)	0	conv_7b_bn[0][0]
=====			
Total params: 54,336,736			
Trainable params: 54,276,192			
Non-trainable params: 60,544			

Figure 28 – Last layers of Inception-ResNet

Thus, observing the complete model, after adding the last fully connected layers in order to adapt to our classification, we have a number of weights to train of 8, comparing to the 492 that we should train if we wanted to fine tune all the model.

Layer (type)	Output Shape	Param #
inception_resnet_v2 (Model)	(None, 4, 4, 1536)	54336736
flatten_1 (Flatten)	(None, 24576)	0
dense_1 (Dense)	(None, 512)	12583424
dense_2 (Dense)	(None, 5)	2565

Total params: 66,922,725
 Trainable params: 66,862,181
 Non-trainable params: 60,544

This is the number of trainable weights before freezing the conv base: 492
 This is the number of trainable weights after freezing the conv base: 8

Figure 29 – Inception-ResNet model adapted to our task

Finally, analysing the figure of accuracy and loss for the model, we can observe a more normal behaviour in this network. What we can appreciate is that the training and validation takes the same tendency, as the training maintains for all the epochs a similar accuracy of 80% and the validation follow the same pattern. Regarding the loss, we can observe even a more similar pattern on the way that the plot behaves. The training achieve a minimum loss of 46% and the validation goes around the same value.

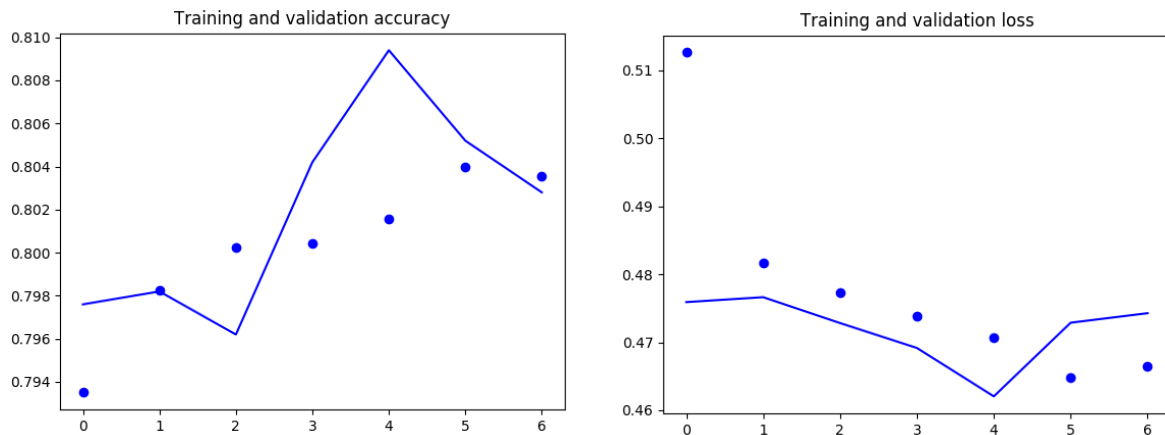


Figure 30 – Accuracy and loss Inception-ResNet model

Another thing that we can extract from this graphic is that in the epoch 4 is when it produces the overfitting, but even that, the variation is not so big so we can consider that the overfitting is quite low.

Even the values of the accuracy and loss are not the best, this model can give us a more adapted result to our database. Thus we can extract more information with it and we could make better predictions more useful for the pre-filter that we want to proportionate to the persons which are doing trademark retrieval.

4.4. Qualitative evaluation of the results

Observing and analysing the results presented in the last section, could be difficult to determine which model is better or which one can give us more information. In order to have a better way to consider the results we present a qualitative evaluation of them.

First of all we would like to remember the different categories we have in the dataset:

- 01 CELESTIAL BODIES, NATURAL PHENOMENA, GEOGRAPHICAL MAPS
- 02 HUMAN BEINGS
- 03 ANIMALS
- 05 PLANTS
- 24 HERALDRY, COINS, EMBLEMS, SYMBOLS

In the next table we have an image from the test directory of the first dataset. We can see in the left part of the table the models which the image have been evaluated and then in the right part the consequent tan percent predictions of every class for each model.


	Class 01	Class 02	Class 03	Class 05	Class 24
Simplest first model (1)	0.17073928	0.16605698	0.17076612	0.19264235	0.22051336
Update first model (2)	0.20914543	0.20788226	0.17503168	0.20515011	0.21575256
VGG16 First Dataset (3)	0.36968872	0.03492463	0.04661944	0.80892724	0.01728138

Table 9 – Predicted percentage of 1st image

Observing the image, without having a really accurate knowledge of the different classifications that can be comprised, we can clearly see that this logo belongs to class 5 which is related to plants. But if we take a look of the predictions of the first two models, we can observe that the category where the image belongs is not clear and the model give a bit higher percentage to class 24 which has nothing in common with class 5. Nevertheless, if we appreciate the third model percentage, we can easily see 81% approximately that pertain to class 5 versus the 36% of belonging to class 1.

This behavior made us think that even the results of accuracy and loss were not bad in general terms for the first and second model, if we go more specifically, we can see that the results are far away to be good. But if we consider the results of the pre-trained network VGG16, is a good beginning to achieve a good model.

Thus, continuing to evaluate qualitatively the results we are going to analyze more images, which would be more difficult to classify than the first logo that we evaluated.



	Class 01	Class 02	Class 03	Class 05	Class 24
Simplest first model (1)	0.24911645	0.23223729	0.24310967	0.27185076	0.29341185
Update first model (2)	0.27377605	0.26483116	0.26037946	0.27783334	0.29498973
VGG16 First Dataset (3)	0.52239341	0.03079538	0.03536591	0.47886807	0.09839684

Table 10 – Predicted percentage of 2nd image

Analysing the image, a person familiarized with the Vienna Classification can relate this logo with the class 1 because of the sun, which belongs celestial bodies, and also because of the drop, which pertain to natural phenomena. This, that could be so easy to interpret for a person, the first and the second model are impossible to determinate, as we can see in the percentages.

But the third model makes us realize that the CNN goes further than a person and it takes more things in count in order to decide between one and another category. We can observe that even the highest percentage is from class 1 with 52%, the model considerate that could belong also to class 5 with a 47%. The conduct that we can extract from that, is that the CNN probably thought about class 5 because we have a green color background and the sun also could be related to plants. So in terms of giving more information to the person who has to classify the new logos is an important point because can help to determine and think in more possibilities.

In the next example we can observe a proof of a logo that is totally an abstraction of the real idea of an animal. We can see a bull because of the silhouette and some typical characteristics of them like the horns or the cue. Thus, we can consider easily in the point of view of a person that is an animal. But normally for a CNN could be difficult to extract the concept.



	Class 01	Class 02	Class 03	Class 05	Class 24
VGG16 First Dataset(3)	0.02126052	0.1381328	0.8736788	0.02168895	0.00135706
VGG16 fine tuning (4)	0.08272404	0.28516683	0.30027634	0.04518246	0.01435558
Inception-ResNet (5)	0.18272278	0.11908115	0.90999109	0.00174844	0.00353586

Table 11 – Predicted percentage of 3rd image

So in this case we discarded the first and the second models, because as we can see the predictions are quite bad and doesn't correspond to the ground truth. Therefore, the idea to take the pre-trained networks of ImageNet that give us the learning of the truth idea of the objects and then transport it to our model has been a good methodology.

Observing the results of the prediction, we can see that the 3rd model and the 5th achieve clearly with a 90% the class what the image belongs, class 3 animals. But the fact that the 4th model don't achieve it with the same clarity it means that it is still an abstract idea and that there's not a 100% of truth that the CNNs can achieve always the best result.

When we are in front of a more complicated trademark image, then we can see better which can be the best model for our database. In the following one, in the way of analyse of a person we won't be so sure where it belongs, because we have the cross and the sphere which can make us think that it could be from the class 24, but also the spherical Earth can make us put it in the first class.



	Class 01	Class 02	Class 03	Class 05	Class 24
VGG16 First Dataset(3)	0.20125511	0.11508962	0.09540103	0.31870216	0.20564628
VGG16 fine tuning (4)	0.40114841	0.08210602	0.08086868	0.29755896	0.1405216
Inception-ResNet (5)	0.84251028	0.02903732	0.04153622	0.07465192	0.20265749

Table 12 - Predicted percentage of 4th image

So in this example the person who has to decide the category of this image can have a lot of doubts between class 1 and class 24. In this case we can see that the more advanced models can give us more information because in case of the Inception-ResNet CNN we have an 84% of probabilities that is class 1. Thus, the more complicated the logo is the more advanced and pre-trained the network.

Finally, if we have to analyze the following image, even an expert in the field of classification could have a lot of doubts for considering which could be the optimum class to belong. Supposing this case, what we prefer in a network is to extract as much information as it can and also give us another way of seeing the logo.

In my personal experiment, when I analyzed this image I was quite convinced that this image has to go to class 24 because it is the shape of a coin. But then after seeing the prediction of the Inception-ResNet model, I realized that in the middle there is a lion that I didn't see it before.



	Class 01	Class 02	Class 03	Class 05	Class 24
VGG16 First Dataset(3)	0.09215027	0.04312395	0.17470659	0.32745904	0.31567219
VGG16 fine tuning (4)	0.12354019	0.12235179	0.13404837	0.12897053	0.45510632
Inception-ResNet (5)	0.14672048	0.17438132	0.40393823	0.08135509	0.33705673

Table 13 - Predicted percentage of 5th image

Thus, even if in this case could be a little bit better the 4th model with the VGG16 Net applying a fine tuning because has 45% of class 24, the other values don't contribute in any other additional information. On the other hand, the Inception-ResNet model could give us a "wrong" result considering more percentage to the class 3 because of the lion, the model give us more information because of the 40% of class 3 and the 33% of class 24.

5. Budget

In this project there is not any aim of creating a prototype to be sold, nevertheless there is a personal dedication cost and some software and hardware used in order to develop the project. But, in future scenarios would be that a production process start and then the cost would be decreased and taking more in count.

This project was carried out in the Alpen-Adria University as a degree thesis so there is not any investment or cost that should count apart from the dedicated time of the persons involved in the project.

Concept	Price per hour	Number of hours	Subtotal
Junior Engineer	15€	750	11250 €
Professor/Engineer	25€	30	750 €
		TOTAL	12000 €

Table 14 – Theoretical personal cost

From the part of software and hardware, we have to distinguish that all the software that we use is Open Source and also the Database we used was Open Data. Thus, the only material that we can count in the budget would be the computer.

The computer for this project has to be powerful with computing and with a potent graphics card. In our case we had a computer with an Intel-i7 processor and an Nvidia GeForce GTX 780 GPU. The cost of that computer would be around 1000€.

Therefore, we can approximate the total cost around **13000€**. This cost would be without taking in count the consequently resources of the office renting, electricity, internet connection...

6. Environmental Impact

The environmental impact of this project is really few in terms of real impact because there is not any final product which can creates pollution or residues. Even that, there is a part of power consumption of the computer used in this project. In some tasks of the project the computer needed to be lots of ours running and computing, so in a really small-scale we can consider a minor impact.

Then we can consider the heat generation of the use of the computers. The use of the computers and technology is responsible for more carbon dioxide emission than manufacturing or disposal. So thinking in a limited size there's a few impact of the dioxide emission, but in our case is negligible.

If we consider the social branch of this project, there is an ethic side that makes you contemplate if it's beneficial or not. In the way that is thought this project is to help the patent offices to generate a more accurate tag when they classify the trademarks. But could be the chance that in the future the classification can be done by a computer program and then work places could be affected.

7. Conclusions and future development:

Finally, this chapter tries to close the thesis exposing the main points of the work done. If we look back to the project idea and proposal, and we review all the parts, we can determine that we have finally achieved the general purpose of the project.

After going through all the previous related works and papers, we can say that we achieved a really good knowledge about the field. We did a notable work with our database, finding the more convenient dataset and extracting the data. We have evaluated the most relevant techniques and algorithms in the actual state of the art. And we had assessed the results which had been quite promising.

Regarding the database, we had more difficulties than the expected to acquire correct labelled data. The process was slower and restricted to fewer data than we had hoped for. This factor that we had fewer data at hand has constrained a little bit the project in order to achieve better scores and be able to try more experiments with it. But even the difficulties, we have reached a really good result with the sparse data we had.

Analysing the models of the convolutional neural network, we can come to the conclusion that the best model we experimented is the Inception-Resnet. We selected this network prioritizing the amount of information that the model gave to us than the concrete results that can achieve with a special case. Thus, we can understand the trend that, going deeper and complex in a model may provide better results and information.

Focusing on the personal aspects, I consider that I learned a lot and increased my skills in autonomous work and solving incoming problems. I think that I have overcome all the difficulties that appear in the path in a correct and easy-going way, and I have been always convinced of my capabilities.

Regarding the possible developments of this project, first of all I would suggest to extract all the labelled data from the OEPM from 1890 to 2017. A second step could be to use the best model extracted from this project, Inception-ResNet, and before applying in our database, do a first fine tuning in a trademark database already created. Then do a second fine tuning of the model in our database and evaluate the results. With the combination of more data and the better CNN model, I encourage to expand the classification task to all the 29 categories of the Vienna Classification, and also incorporate the most relevant subcategories of each class.

Bibliography:

- [1] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In *Advances in Neural Information Processing Systems 25*, pages 1106-1114, 2012.
- [2] Matthew D. Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In David J. Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, 'Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I, volume 8689 of Lecture Notes in Computer Science, pages 818–833. Springer, 2014.
- [3] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". arXiv preprint arXiv:1409.1556, 2014.
- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–9, 2015.
- [5] K. He, X. Zhang, S. Ren, and J. Sun. "Deep residual learning for image recognition". arXiv preprint arXiv:1512.03385, 2015.
- [6] Christian Szegedy, Sergey Loffe, Vincent Vanhoucke and Alex Alemi. "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning" arXiv preprint arXiv: 1602.07261, 2016.
- [7] Ke Zhang, Miao Sun, Tony X.Han, Xingfang Yuan, Liru Guo and Tao Liu. "Residual Networks of Residual Networks: Multilevel Residual Networks". arXiv preprint arXiv: 1608.02905, 2017.
- [8] Yann LeCun, Yoshua Bengio and Geoffrey Hinton. "Deep Learning Nature". Review doi: 10.1038/nature 14539. 2015
- [9] Hang Su, Shaogang Gong, Xiatian Zhu. "Scalable Deep Learning Logo Detection". arXiv preprint arXiv: 1803.11417, 2018.
- [10] Keras Documentation: Available at <https://keras.io/>
- [11] François Chollet. (2017). *Deep Learning with Python*. MEAP Edition Version 5. www.manning.com
- [12] François Chollet. Xception: "Deep Learning with Depthwise Separable Convolutions". arXiv preprint arXiv: 1610.02357, 2017

Glossary

- CNNs – Convolutional Neural Networks
- CPU – Central Process Unit
- GPU – Graphics Processing Unit
- WIPO - World Intellectual Property Organization
- VCL – Vienna Classification
- XML – Extensible Markup Language
- OEPM – Oficina Española de Patentes y Marcas
- AI – Artificial intelligence
- DL – Deep Learning
- ILSVRC – ImageNet Large-Scale Visual Recognition Challenge