UNIVERSITAT JAUME•I

Deep Learning Techniques Applied to Videogames

Víctor van der Kolk Álvarez

Final Degree Work

Bachelor's Degree in
Video Game Design and Development

Universitat Jaume I

July 16, 2021

Supervised by: Raúl Montoliu Colás.

To Rosa and Jan

Acknowledgments

First of all, I would like to thank my family for the unconditional support they have given me throughout my life both personally and academically, allowing me to fulfill my dreams without holding me back at any time.

Secondly, I would like to mention all those people who have accompanied me throughout these 4 years of my career, whether they have stayed by my side or not, because they have always managed to bring something new to my life.

Thirdly, I would like to thank my supervisor Raul Montoliu, for guiding me throughout the development of the project for its proper preparation and advising me throughout the race when I had any doubts outside his field.

Finally, I would like to highlight the importance of each of the professors that I have had the pleasure of being taught by, for teaching me the different subjects in which they have specialized; and in particular to Sergio Barrachina Mir and José Vte. Martí Avilés for their inspiring LaTeX template for writing the Final Degree Work report, which I have used as a starting point in writing this report.

Abstract

This document is a reflection of the work done as a final degree project.

In summary, the work has consisted of research on techniques capable of generating and simulating artificial intelligence in general, and in particular, learning to use deep learning techniques through the use of neural networks.

The project has been divided into three different parts: research about artificial intelligence and its techniques, testing of existing examples in order to learn how to develop related techniques, and application of the previous concepts to a specific topic related to videogames using the previous techniques.

Thus, throughout the report each part of the process will be explained in detail following the above order and emphasizing the most important points, starting with the technical explanation of what a neural network is and what it is used for; following with the different examples that have been followed to understand and make use of them; and ending with the development process of a deep learning project capable of estimating the total duration of a game in order to evaluate whether the pairing of the players that form it has been correct or not.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

Contents

Throughout this chapter, the main reason that has led to the development of the project will be explained, what are the main objectives to be met and what is expected to be obtained from it.

1.1 Work Motivation

After having designed and programmed video games or applications where artificial intelligence has not had a main role in the project and due to the importance that programming techniques and algorithms related to deep learning are currently gaining, I see this project as a great opportunity to introduce myself to this field of software development.

On the other hand, in no subject of the degree is explained in detail artificial intelligence techniques developed with neural networks such as machine learning or deep learning. At most, these types of techniques are superficially mentioned in the subjects of Artificial Intelligence (AI) and Advanced Interaction Techniques (AIT), which I think is a pity, because recent studies of both video games and computer science, in general, have shown that when learning to develop software, these concepts are fundamental.

That is why the main reason for this project is to learn as much as I can about neural networks, machine learning, and deep learning. There is no actual need to have perfect results on the different projects I am going to test out and develop since the objective is not to program the perfect neural network model but learn all I can about them by following already implemented examples and trying to develop my neural network in a problem related to videogames' field.

1.2 Objectives

The main objectives of the project can be listed as follows:

- Understand in general terms how neural networks work.
- Puzzle out which are the different existing deep learning techniques and what are they used to.
- Study how problems related to video games can apply these techniques.
- Learn how to use the main deep learning techniques.
- Design a problem related to video games and provide a solution to it by applying what has been learned in the previous steps

1.3 Environment and initial state

To put the reader in context, this project starts from a very light idea about what a neural network is and practically zero certainties about the different applications and techniques that have led to their development. For the development of this project, the only prerequisite was to maintain a relationship between the main topic and video games, therefore, it was decided to take the first approach as learning about current deep learning techniques and a second approach that starts from current real applications and a possible application within the world of videogames.

1.4 Expected results

As said above and keeping the project objectives as expected results, what is intended to be achieved in a generalized way with this project is:

- More extensive knowledge about what deep learning is.
- How to use techniques related to deep learning.
- Understand what an API is and make use of some public ones.

- Solve a problem related to the world of video games by applying deep learning.

The problem that it has been decided to tackle using deep learning techniques is a commonly known problem in online multiplayer video games: player matching or commonly known as matchmaking.

Matchmaking is a system that tries to match players or teams of players of a similar level or skill by analyzing their statistics from previous games to establish the rank of each player [15].

Chapter 2

Planning and resources evaluation

Contents

This chapter will serve to define the planning and costs that have been taken into account throughout the project. For this reason, firstly, the times that have been set for each phase of the project will be detailed, and secondly, the real-time costs that have been necessary for its development.

2.1 Project schedule

For the planning of this project, it has been decided to distinguish between document writing, research and learning, and personal development when allocating the time necessary for the correct fulfillment of each phase of the project. It should be noted that the hours assigned are an arbitrary estimate of the duration of each task concerning the total estimated for the project, 300 hours.

Firstly, Table 2.1 the different tasks of the project related to its publication can be distinguished in terms of writing and presentation.

Table 2.1: Documentation phase's estimation

Documentation phase	
Tasks	**Estimated duration (in hours)**
Technical proposal	5
GDD proposal	5
Technical report	40
Project defense presentation	10
Total	**60**

Secondly, Table 2.2 the tasks related to research and learning about the main topic of the project have been considered: neural networks. This phase of the project focuses mainly on the search for information and understanding of the subject.
Furthermore, it also takes into account testing and following already implemented models about image identification such as classify between cats and dogs or image manipulation like face detection or resolution changes.

Table 2.2: Pre-development phase's estimation

Pre-development phase	
Tasks	**Estimated duration (in hours)**
Research about what a neural network is	10
Investigate how neural networks are implemented	10
Learn the main concepts about what an API is	10
Test already implemented neural network models	20
Follow tutorials and develop some guided examples	100
Total	**150**

Thirdly, Table 2.3 shows those tasks related to the development of neural networks and video games. Within these points, tasks such as data manipulation and downloading are also taken into account.

Table 2.3: Development phase's estimation

Development phase	
Tasks	**Estimated duration (in hours)**
Design a solution for the proposed problem	5
Find a public video game database with data of different matches	5
Understand and learn to use an API related to that database	10
Develop a neural network in order to predict the duration of a game	60
Test and evaluate the project	10
Total	**90**

Finally, and to outline the project, the different sections that are going to make up its development are briefly and concisely presented in Table 2.4.

Table 2.4: Estimated duration of project's tasks summary

Summary table	
Tasks	**Estimated duration (in hours)**
Documentation	60
Pre-development	150
Development	90
Total	**300**

2.2 Project execution and actual cost

This section has as its main objective to compare the previous assessment with the real costs that the project has implied in durations terms after evaluating the necessary resources needed to develop the project in Tables 2.1, 2.2 and 2.3. To achieve this aim, the project tasks and the dependencies that exist between themselves have been outlined as a Gannt chart represented in Figure 2.1, so it is easy to observe how the project has been developed over time.

As it can be seen in Figure 2.1, there is a variation between the real duration of the tasks of the project and the estimated ones. This is due to the appearance of some issues along the development process.

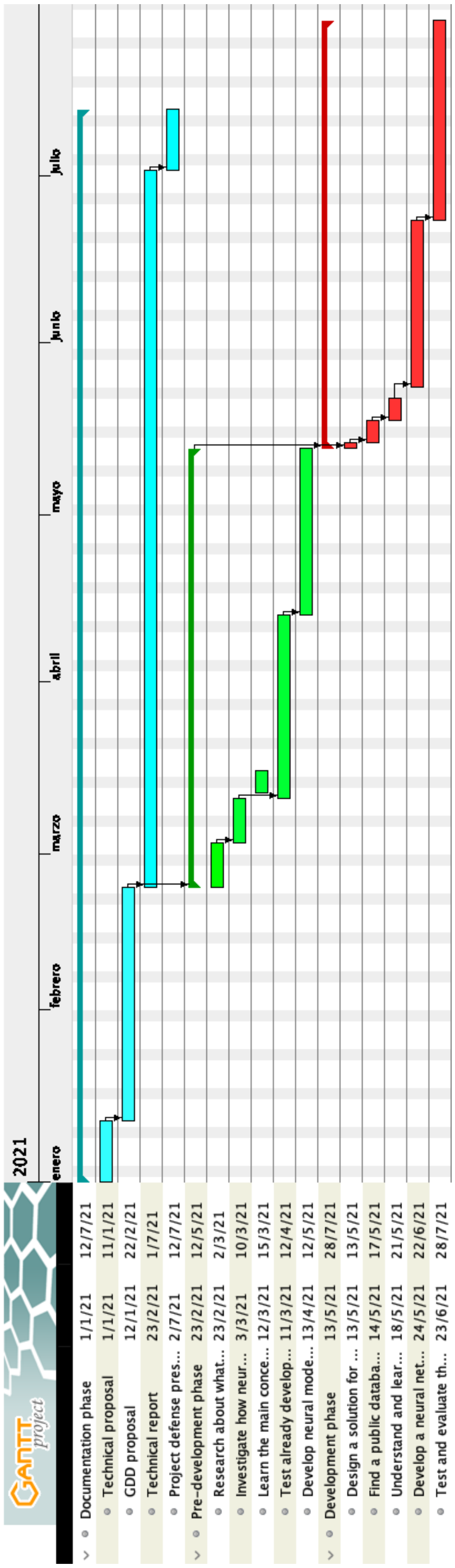
On one hand, the main problem of the project has been the ambiguity of how to deal with matchmaking in video games and how to reproduce a neural network able to identify if a match has been correctly balanced by analyzing the duration of the game by processing the skill level of each player of the game and if they were in a hot streak (3 wins in a row or more).

On the other hand, when preparing the dataset to train the neural model, little prior knowledge about how a dataset is created has been an important factor to take into account when delaying the process of tasks.

Finally, in order to evaluate the economic aspect of the project, it can be seen in Table 2.5 that a cost evaluation has been carried out taking into account the computer used, the programs that have allowed the development of the project and the labor force based on an average hourly salary of a common software developer.

Table 2.5: Economic costs of replicating this project by a company hiring one employee

Cost evaluation summary			
Resource	**Cost (in €)**	**Time spent (in hours)**	**Total (in €)**
Hardware costs			
Computer	2949.49		2949.49
Graphic card	995.99		995.99
Software license costs			
Windows 10 Pro	109.90		109.90
PyCharm	199.00 / year	5096 (0.58 years)	199.00
GitHub	4.00 / month	5096 (7 months)	21.00
Human costs			
Employee	22.00 / hour	364	8008.00
Total			**12283.00**



Documentation phase	1/1/21	12/7/21
Technical proposal	1/1/21	11/1/21
GDD proposal	12/1/21	22/2/21
Technical report	23/2/21	1/7/21
Project defense pres...	2/7/21	12/7/21
Pre-development phase	23/2/21	12/5/21
Research about what...	23/2/21	2/3/21
Investigate how neur...	3/3/21	10/3/21
Learn the main conce...	12/3/21	15/3/21
Test already develop...	11/3/21	12/4/21
Develop neural mode...	13/4/21	12/5/21
Development phase	13/5/21	28/7/21
Design a solution for ...	13/5/21	13/5/21
Find a public databa...	14/5/21	17/5/21
Understand and lear...	18/5/21	21/5/21
Develop a neural net...	24/5/21	22/6/21
Test and evaluate th...	23/6/21	28/7/21

Figure 2.1: Gantt chart of the project (made with GantProject)

Tables 2.6, 2.7 and 2.8 display the real duration values in order to show how the project has undergone slight modifications depending on the needs found in each task.

Table 2.6: Documentation phase's duration

Documentation phase		
Tasks	**Estimated duration (in hours)**	**Real duration (in hours)**
Technical proposal	5	6
GDD proposal	5	4
Technical report	40	67
Project defense presentation	10	8
Total	**60**	**75**

Table 2.7: Pre-development phase's duration

Pre-development phase		
Tasks	**Estimated duration (in hours)**	**Real duration (in hours)**
Research about what a neural network is	10	8
Investigate how neural networks are implemented	10	12
Learn the main concepts about what an API is	10	7
Test already implemented neural network models	20	23
Follow tutorials and develop some guided examples	100	138
Total	**150**	**188**

Table 2.8: Development phase's duration

Development phase		
Tasks	**Estimated duration (in hours)**	**Real duration (in hours)**
Design a solution for the proposed problem	5	4
Find a public video game database with data of different matches	5	9
Understand and learn to use an API related to that database	10	13
Develop a neural network in order to predict the duration of a game	60	71
Test and evaluate the project	10	4
Total	**90**	**101**

Finally, and to outline the differences between the estimation and the actual duration, the different sections are, once again, briefly and concisely presented in Table 2.9.

Table 2.9: Real duration of project's tasks summary

Summary table		
Tasks	**Estimated duration (in hours)**	**Real duration (in hours)**
Documentation	60	75
Pre-development	150	188
Development	90	101
Total	**300**	**364**

2.3 Tools

This section lists the tools that will be used during the development on this project. They are grouped according to the purpose for which they have been used.

- **Documents**
  - **TexShop:** free local platform for scientific writing based in LaTex.
  - **TablesGenerator:** free online web for creating Latex tables.
  - **KeyNote:** free Apple app to create graphic presentations.
  - **GanttProject:** free open app to create Gantt diagram projects.
  - **Grammarly:** free online app that corrects grammatical mistakes.

  - **DeepL:** free online translator.

- **Programming**
  - **Anaconda:** open source, flexible solution that provides the utilities to build, distribute, install, update, and manage software in a cross-platform manner.
  - **PyCharm:** python development environment.
  - **Riot Games API:** public videogame's API that allows getting access to League of Legends data.

- **Version control**
  - **GitHub:** a Git repository hosting service.

Chapter 3

Research and documentation

Contents

This chapter serves as a learning introduction to the different terms that will be covered throughout the project. Therefore, following the logical order of things, the chapter will start from what artificial intelligence is to which different deep learning techniques are applied to videogames these days.

3.1 Artificial Intelligence

Artificial intelligence (AI) refers to the simulation of human intelligence in machines that are programmed to think like humans and mimic their actions. The term may also be applied to any machine that exhibits traits associated with a human mind such as learning and problem-solving [14]. Put more colloquially, artificial intelligence is defined as the attempt to simulate or imitate human cognitive functions (thinking, reasoning, learning, etc.) by a machine.

It is important to differentiate between the terms artificial intelligence, machine learning, and deep learning. On the one hand, AI, as already explained, is the ability of computers to display intelligent behavior, while on the other hand, machine and deep learning are nothing more than different learning techniques to obtain said simulation.

Figure 3.1 shows the main differences between the three main concepts that will be discussed in this documentation section: Artificial Intelligence, Machine Learning, and Deep Learning. As can be seen, deep learning is an evolution of machine learning and these two are nothing more than techniques applied to the generation of artificial intelligence in computer programs.

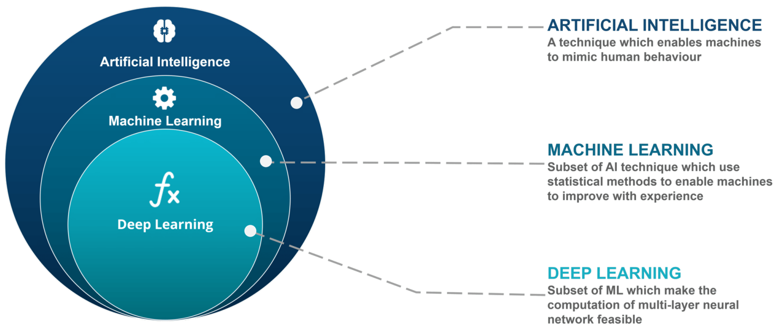

Artificial Intelligence
Machine Learning
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Figure 3.1: Differences between IA , ML and DL

Source: https://www.edureka.co/blog/wp-content/uploads/2018/03/AI-vs-ML-vs-Deep-Learning.png

3.2 Neural networks

A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates [5].

Figure 3.2 represents how a simple neural network is made up of different elements:

Input Layer: a set of different values given as a starting point for each relevant data of the problem to be solved. A neural network can work from 1 to n inputs that provide information from the outside world to the network.

Weight: value associated with each input data and adjustable at any time that acts as a link between the input layer and the hidden layer.

Hidden Layer: performs all sorts of computation on the features entered through the input layer and transfer the result to the output layer. Internally, it calculates a sum of the inputs multiplied by each of the weights associated with them, recreating a linear regression model[35] and then using an activation function.

Output Layer: it is the result of the calculation of the previous layer.

Bias: is a constant value that is used as the effect of shifting the activation function, in other words, the line is effectively transposed by the constant value.

Activation Function: mathematical function that allows modifying the linear regression model to shape curves or areas within the model instead of a line.

Figure 3.2 shows a simplified representation of a single-layer neural network, separating the different parts of the network into an input layer, associated weights, the activation function [37] (which is binary in this case), and finally, the output layer with the result obtained.

.

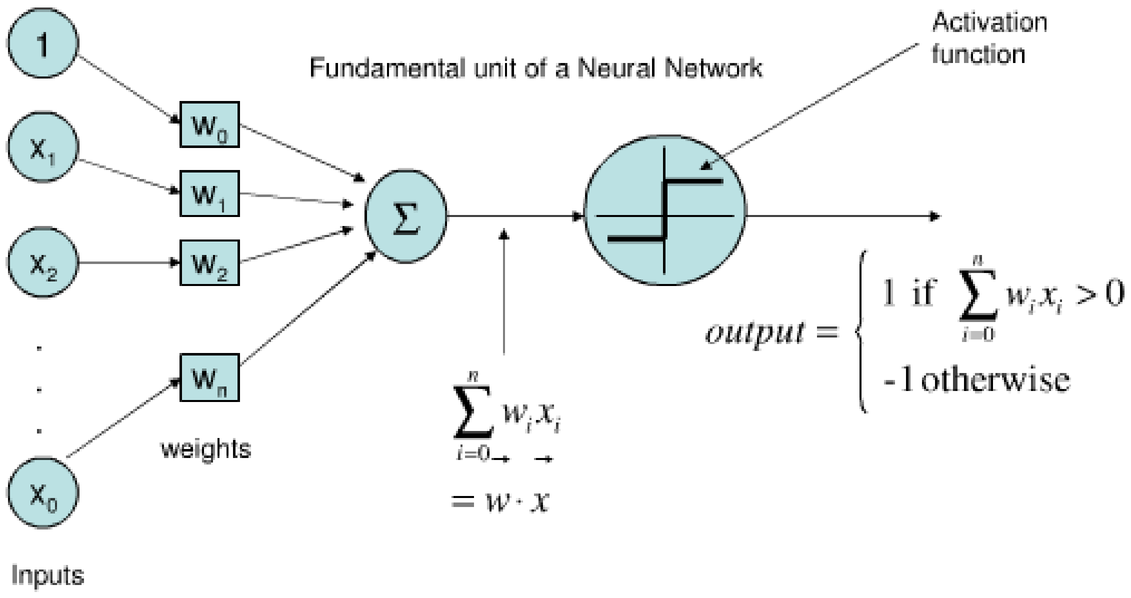

Fundamental unit of a Neural Network
Activation function
1
X_1
X_2
X_0
Inputs
W_0
W_1
W_2
W_n
weights
Σ
$\sum_{i=0}^{n} w_i x_i = \vec{w} \cdot \vec{x}$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Figure 3.2: A Single Layer Neural Network

Source: https://www.programmersought.com/images/258/633dca25508a646a6df343339c3d4eaa.png

3.3 Machine learning

Machine Learning can be defined as a "field of Computer Science that gives computers the ability to learn without being explicitly programmed" - Samuel Arthur, 1969. Another way of defining machine learning is the act of computer learning to recognize patterns from a set of data.

Supervised learning: these are generally classification or regression problems and a sample input/output data set is established to train the model. It consists of two steps: learning and testing.
The learning stage consists of entering a set of previously analyzed data manually

known as featured engineering so that the model can readjust the internal parameters that compose it to obtain a predefined result.
In the second, once the model has been trained, new data is introduced and this is capable of generating a prediction of a possible solution to the problem with enough precision from the previous training.

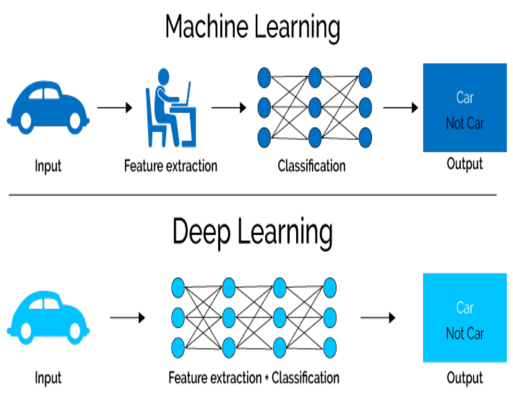
Unsupervised learning: the input data set does not have a predefined output and therefore it is the model which is responsible for finding a relationship between the different data. This process is common in clustering problems.

Semi-supervised learning: something intermediate between the two previous points. They are common in problems where you don't have as many example outputs as inputs.

Reinforcement learning: it doesn't have specific example results but goals are set through a reward system. The model based on trial and error modifies its parameters to obtain the highest possible reward.

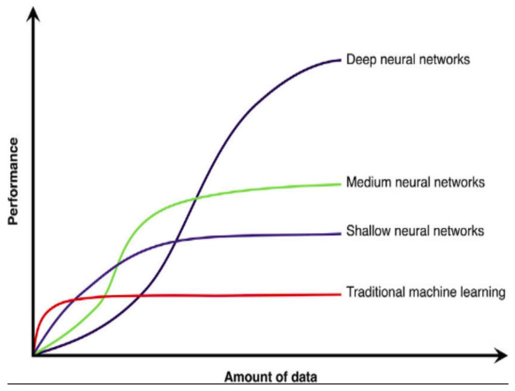
3.4 Deep learning

Deep Learning is an evolution of Machine Learning that specializes in emulating the human brain from a computer. It is characterized and differs from ML in that it is the algorithm itself that is capable of analyzing the relevant characteristics of the data entered instead of having to do it manually previously, and therefore, as the model is trained, it is capable of learning how to extract the characteristics of the data that are useful to you. In short, deep learning eliminates the manual part of machine learning by automating the process in exchange for increasing the input data set (3.3a). Thus, deep learning algorithms perform better the more data they have when training the model as shown in Figure 3.3b.



(a) ML vs DL stages

Source: https://lawtomated.com/wp-content/uploads/2019/04/MLvsDL.png



(b) DL vs ML performance

Source: https://www.researchgate.net/figure/Graph-illustrating-the-impact-of-data-available-on-performance-of-traditional-machine_fig1_324457640

Figure 3.3: Main differences between Machine and Deep learning

The main difference between these two forms of Artificial Intelligence is that in machine learning, you have to guide the machine in each of the phases of the process so that it learns, through practice, to identify what we want automatically. Conversely, in deep learning, the machine learns by itself with each new input of information it receives. If you ever use the wrong piece of data, learn from the mistake and use other data to get closer to the correct result faster and more reliably so that failure will never happen again [2].

3.4.1 Types of DL techniques

For the preparation of this section, the article *Top 10 deep learning algorithms you should know in 2021* has been used as reference and inspiration [3], but only some of these techniques will be explained due to their interest and importance for the development of this project.

Multilayer Perceptrons (MLPs)

Probably the best starting point for understanding a multilayer neural network. Internally, it has a structure similar to a simple neural network, unlike that, it is composed of more than one hidden layer, that is, the output of each hidden layer is redirected to the next hidden layer linearly until the final output is reached. Figure 3.4 shows a simple example of how a multilayer neural network works in a schematic way in order to distinguish between dogs and cats. First, the user provides the network with a feature vector in order to classify each image. After that, the network adjusts the different weights associated with each layer until a result is obtained where the final label of the vector corresponds to the classification obtained. Thus, the neural network is now able to distinguish between dogs and cats through a provided feature vector.

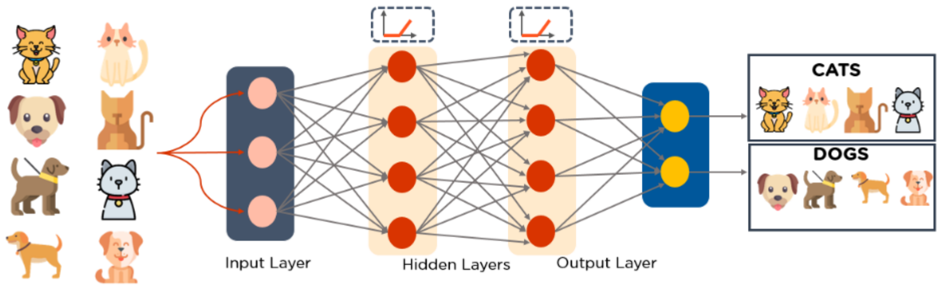


Figure 3.4: MLP diagram

Source: https://www.simplilearn.com/ice9/free_resources_article_thumb/Multilayer_Perceptron.PNG

Convolutional Neural Networks (CNNs)

Consists of the use of different layers capable of processing and detecting similarities in an image. It is very useful when classifying and/or detecting objects in an image. Unlike the multilayer neural network, as shown in the figure 3.5, the network receives as input directly the image and is responsible for readjusting the parameters of each layer to obtain the similarities of each image. In other words, the numbers that allow to obtain the desired results are learned autonomously by the network, the feature vector is not necessary.

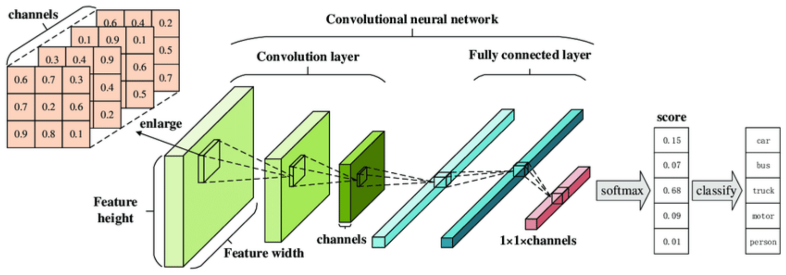
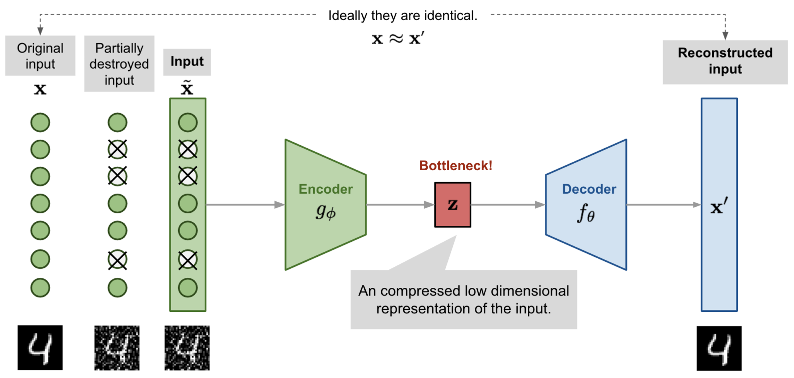


Figure 3.5: CNN example

Source: https://www.researchgate.net/publication/330106889/figure/fig1/AS:710963951063040@1546518423301/Architecture-of-a-Convolutional-Neural-Network-CNN-The-traditional-CNN-structure-is.png

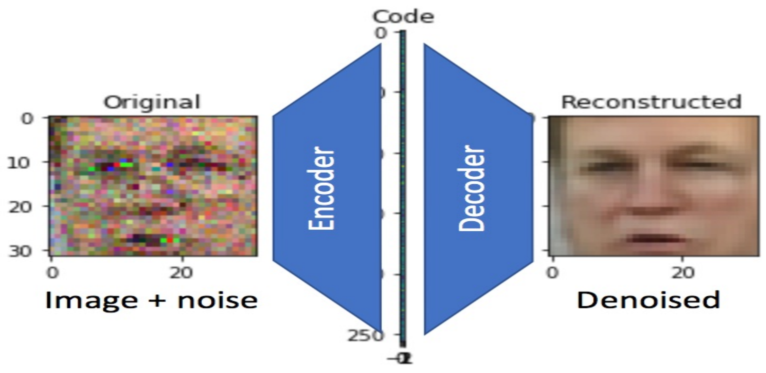
Autoencoders

It is generally used in the recreation of unsharp images by encoding and decoding a blurred image. A very common example is character correction when writing on a graphics tablet with a ballpoint pen. Thus, Figure 3.6 represents the inner workings of this type of model. First, the image parameters are encoded in order to obtain representative values of the image, something similar to a CNN. However, unlike CNNs, the next step is to send these values to a decoder in order to obtain a result similar to the original but modified according to the needs for which the network has been trained, such as increasing or reducing the resolution of the original image.


Ideally they are identical.
$\mathbf{x} \approx \mathbf{x}'$
Original input
$\mathbf{x}$
Partially destroyed input
Input
$\tilde{\mathbf{x}}$
Encoder
g_ϕ
Bottleneck!
$\mathbf{z}$
Decoder
f_θ
Reconstructed input
$\mathbf{x}'$
An compressed low dimensional representation of the input.

(a) Handwritted digit reconstructed

*Source: https://miro.medium.com/max/777/1*ce89U6z-MhgGRln9VRAMxQ.png*



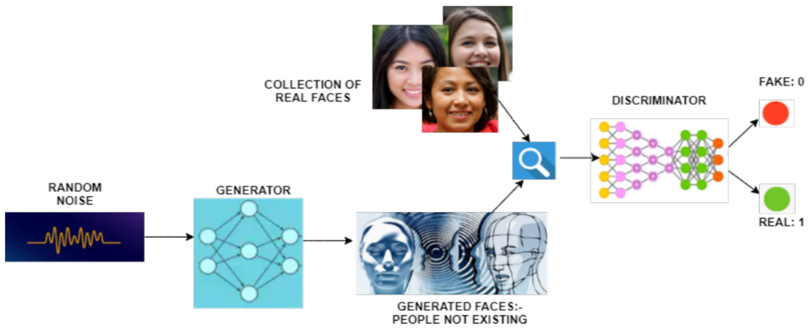
(b) Image denoising

*Source: https://miro.medium.com/max/1030/1*WrBBqd9whs2Xpl5yROYFKw.png*

Figure 3.6: Visual applications of autoencoders

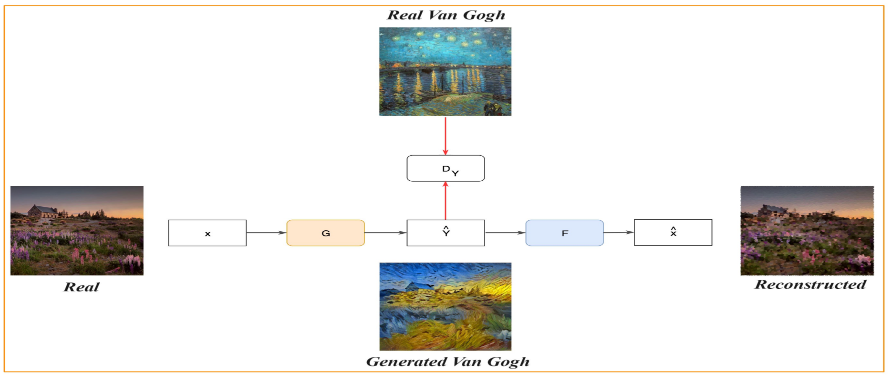
Generative Adversarial Networks (GANs)

They are characterized by having a false model generator and a discriminator that detects when the data to be evaluated is true or false. After evaluating the case, both the generator and the discriminator are updated and retrieved. This constant updating of the model allows that within the world of video games, the resolution of old games can be increased or textures of 3D models can be rescaled. Some examples of this technique can be seen in Figure 3.7.



(a) Gan discriminator model

*Source: https://miro.medium.com/max/1121/1*k6FxInYluTSKeFraFtvIyQ.png*



(b) Gan reconstruction model

Source:

https://www.programmersought.com/images/650/3fbed217f1e824cf4a79f9e49acb7fca.JPEG

Figure 3.7: Examples of GANs' applications

Long Short Term Memory Networks (LSTMs)

The practical exemplification of recurrent neural networks. This type of neural network is commonly used in speech recognition because they allow data to be retained over time while new ones are entered. They work following three steps: forget irrelevant parts of the previous state, selectively update the cell-state values and output certain parts of the cell state.

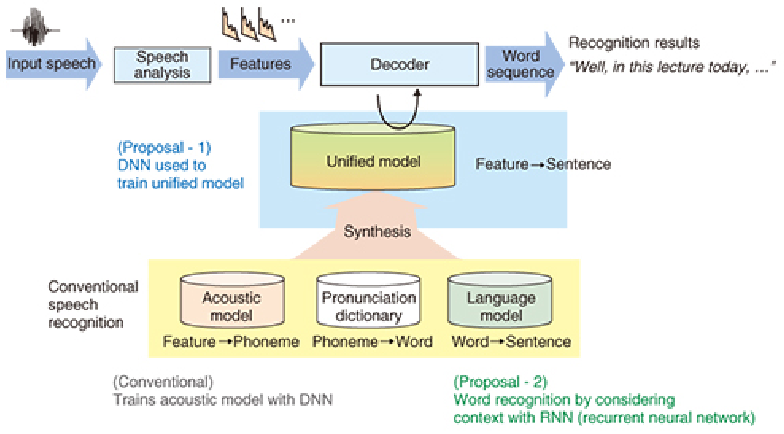

Input speech
Speech analysis
Features
Decoder
Word sequence
Recognition results
"Well, in this lecture today, ..."
(Proposal - 1) DNN used to train unified model
Unified model
Feature→Sentence
Synthesis
Conventional speech recognition
Acoustic model
Pronunciation dictionary
Language model
Feature→Phoneme
Phoneme→Word
Word→Sentence
(Conventional) Trains acoustic model with DNN
(Proposal - 2) Word recognition by considering context with RNN (recurrent neural network)

Figure 3.8: LSTM voice recognition example

Source: https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcSKNzkqmoCRqLSu-qYgL5tdtoPOsV4TbSV7AQ&usqp=CAU

3.4.2 DL in videogames

Within the world of videogames, possibly the most important application of deep learning techniques resides in the infinite improvement of the artificial intelligence of NPCs through reinforcement-based learning techniques. But as has already been exemplified in the previous section, many other applications are possible such as the reconstruction of textures, 3D models, audios, etc. of old video games.

It has been shown that at times, different deep learning techniques have surpassed human abilities in different fields such as AlphaGo [40], in strategy games, or GauGan [36] in the artistic world.

On the other hand, when analyzing player statistics where it is necessary to categorize thousands and thousands of data from millions of players, these types of techniques are very useful in problems such as bringing together players of the same level or who have a similar ability that is established by a number.

Finally, augmented reality games or virtual reality games are becoming more and more possible due to deep learning algorithms that allow us to bring the outside world to the coded world that the machine can understand.

That said, throughout the rest of this section different examples where AI has matched or, in some cases, surpassed human intelligence will be described.

Deep Blue, AI vs. human brain in chess games

Deep blue [41] is a supercomputer developed by IBM that became the first computer capable of beating a human in a game of chess. Although it does not use deep learning techniques, it was the beginning of the rise of artificial intelligence against the human brain.

The algorithm used to analyze the state of the game and execute each move in each turn is an algorithm called MiniMax [43] which is responsible for minimizing the maximum expected loss in opponent games and taking into account that it has the information of all the game states. In other words, choosing the best move taking into account that the opponent will choose the worst for you.

OpenAI Atari, Bots against arcade games

Atari 2600 is a video game console from Atari that was released in 1977. The game console included popular games such as Breakout, Ms. Pacman and Space Invaders. Since Deep Q-Networks were introduced by Mnih et al. in 2013, Atari 2600 has been the standard environment to test new Reinforcement Learning algorithms. Atari 2600 has been a challenging testbed due to its high-dimensional video input (size 210 x 160, frequency 60 Hz) and the discrepancy of tasks between games [11].

That said, the main function of this project was to visualize real applications of deep reinforcement learning in world-known games. To do so, the Open AI team used the environments provided by The Arcade Learning Environment (ALE) [26] in order to provide curious programmers with real learning environments for intelligent agents.

For the correct functioning of these trainings, the emulator's customization layer allows the separation between the agent's decisions and the rest of the environment, so that at each tick or frame of the game the agent can perform a new sequence of random actions.

In this way, iteration after iteration and with a reward and error system, the different parameters of the neural network are readjusted in order to obtain the best movement in each state of the environment.

OpenAI Five, 5 bots vs. 5 professional players

Another famous example of deep learning techniques inside the world of videogames is provided by OpenAI company [28]. This company has developed different AI examples using deep learning techniques and two of their most famous projects are related to video game applications.

A project with a similar aim as the previous one, to win human players in videogames, is their project named OpenAIFive [29]. This project consists of the development of an AI able to win a game of Dota2 game. While common bots are easy to defeat since they are scripted to win other simple bots to level up different accounts, the OpenAI team tried to go further by using Valve's BOT API, which gives current information about the game state. For it, they managed to introduce this provided information in 5 different mono-layer LSTM neural networks, simulating the 5 different players a Dota2 team has.

"OpenAI Five averages around 150-170 actions per minute (and has a theoretical maximum of 450 due to observing every 4th frame). Frame-perfect timing, while possible for skilled players, is trivial for OpenAI Five. OpenAI Five has an average reaction time of 80ms, which is faster than humans" [27]. Once again, the power of an AI defeats human skills.

Another one, but no less important, was a Hide and Seek MultiAgent game. The importance of this project resides in how the agents were able to make moves that even the developers did not expect [30].

As before, the first example demonstrates how far a machine can go by not only winning a human player but not letting him the chance to win even a single game unless is forced to it. The second one, replies what people do since they are born, study the environment, think, and act getting good results. It is probably one of the best examples of self-learning and how deep learning has revolutionized the AI world.

DeepMind, AlphaGo

The game of Go is a strategy table game where the outcome of the game solely depends on the strategy of both players, and as it is possible to rely on a machine to find the optimal sequence of moves, it is an attractive problem to solve computationally.

It has always been hardware, and in a minor way, a software problem the amount of different possible moves that are possible to do to win a game. The importance of deep learning in this project resides in how this AI developed Google's Deep Mind research team [10] using already known algorithms boosted by neural networks, was the first software able to win a professional Go player [40].

To solve the problem of how a machine can win a human player, and even further, not get ever defeated by a professional player, Google developed this AI by using a variable of Monte Carlo tree search [44] combining two different neural networks with it. While one neural network takes in the policies of the game (rules, possible moves, etc.) the other one works with the different values that a move can produce, learning where to move every turn (see Figure 3.9 for visual example on how the complete algorithm works).

The interesting part of Alpha Go is how these two neural networks were trained. In the first instance of the problem, the software was trained by using supervised learning aiming to mimic human movements by attempting to match expert moves from recorded games of professional players. But in a second instance, when the machine was already able to mimic those moves, it was trained against different instances of itself using reinforcement learning.

So summing up, the way Alpha Go was trained was by looking up different games of pro players (pattern recognition), learning from different games using supervised learning (policy network in a first instance), and playing against itself by using reinforced learning in both networks.

As it can be seen in the figure 3.9, the algorithm consists of four different parts:

- Pick the optimal move
- Calculate P from the policy network. Return to the previous step
- Compute Q by averaging over the value network AND rollout
- The most visited move is chosen

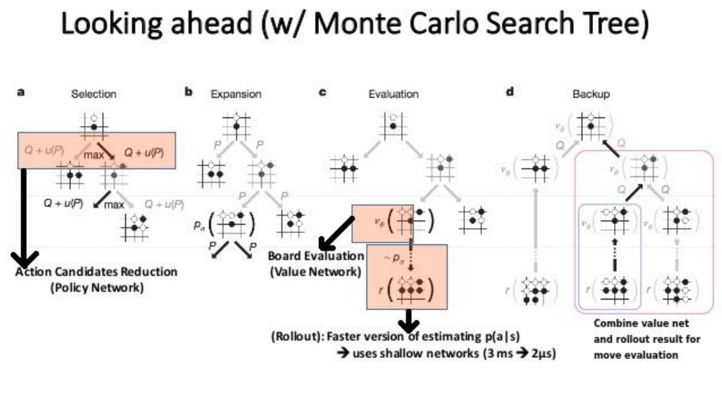

Looking ahead (w/ Monte Carlo Search Tree)
a Selection
b Expansion
c Evaluation
d Backup
Q + u(P)
max
Q + u(P)
Action Candidates Reduction (Policy Network)
Board Evaluation (Value Network)
(Rollout): Faster version of estimating p(a|s)
→ uses shallow networks (3 ms → 2µs)
Combine value net and rollout result for move evaluation

Figure 3.9: How AlphaGo algorithm works

Source: https://www.andreykurenkov.com/writing/ai/a-brief-history-of-game-ai-part-3/

Nowadays, in order not to humiliate the different opponents that the algorithm faces, the victory percentage can be manipulated, thus allowing the human player the possibility of winning the game. This way, the power of deep learning techniques applied to videogames is demonstrated.

DeepMind, AlphaStar

AlphaStar [9] is an artificial intelligence developed, once again, by Deep Mind [10] research team that became famous after it defeated a professional team in a 5vs5 multiplayer strategy video game.

Unlike the previous AlphaGo [40] project, in this case, the AI was limited to the same actions that a human player can have, i.e. a limited amount of actions and reduced visibility of the map.

Again, AlphaStart, like its predecessor AlphaGo, uses deep learning techniques along with reinforcement learning techniques, but the main difference is that the exploration process initially learns to mimic the best player it has in its database and from there plays thousands of games against itself, whereas AlphaGo analyzed thousands of games from different players in order to learn and recognize movement patterns.

NvidiaGauGan

Even though it is not directly related to the world of video games, one of the most impressive technologies at first glance that apply deep learning techniques and in particular the Gan neural network model is the GauGan Nvidia software.

By drawing an abstract form selecting a different type of brushes which determine the to expected figure to be drawn, and passing this information to the GAN model by differentiating the type of input by the color of the pixel of the image the network is processing, it is capable of generating a landscape in high resolution and with great detail quality.

Although its main application is not located within video games, the applications of this type of network when rescaling the resolution of landscapes are numerous. That is why it has been decided to mention this Nvidia technology [36] in this subsection. An example of what this Gan network is capable of can be seen in the following figure 3.10

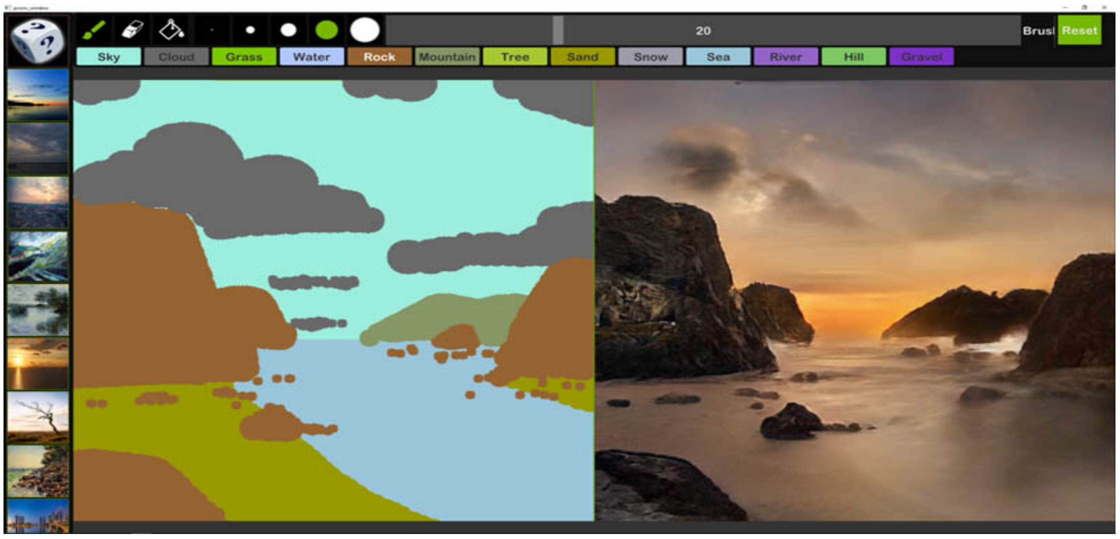


Figure 3.10: Nvidia GauGan Example

Source: https://www.servethehome.com/nvidia-gaugan-perhaps-the-coolest-gtc-2019-demo/

Chapter 4

Work Development

Contents

Throughout this chapter we will differentiate between the different stages that the project has undergone for its correct development. Thus, it has been decided to separate the work carried out in four different sections, taking into account the level of knowledge before starting the section and its purpose.

It should be noted, once again, that the purpose of this project is to investigate a subject outside the field of video games, neural networks, but to relate it to the degree to which this project is presented, it has been decided to take into account some problems that may arise when innovating in the programming of a video game or its mechanics.

4.1 Testing of real applications with pre-programmed and pre-trained neural network models

Throughout this section, knowledge of neural networks and deep learning techniques was only theoretical. Therefore, the main objective of this section is to see real applications without full access to their code.

4.1.1 OpenAi Gym

The first real contact with neural networks in this project has been to test the operation of a pre-trained model.

In this way, the Open AI Github public repository [1] has been accessed, where a battery of models and examples that can be easily tested to see how they work is made available to anyone.

The algorithms that Open AI makes publicly available use deep reinforced learning techniques that work in such a way that at each stage of the algorithm the parameters of the neural network are readjusted to obtain the maximum reward and the least possible punishment.

Some examples of the functioning of these networks can be seen in Figure 4.1 where two different networks have been tested: the first one teaches an animal to walk stage after stage and the second one learns to play one of the most famous games in history, SpaceInvaders.

All of these examples can be downloaded and tested from the following link https://github.com/openai/gym

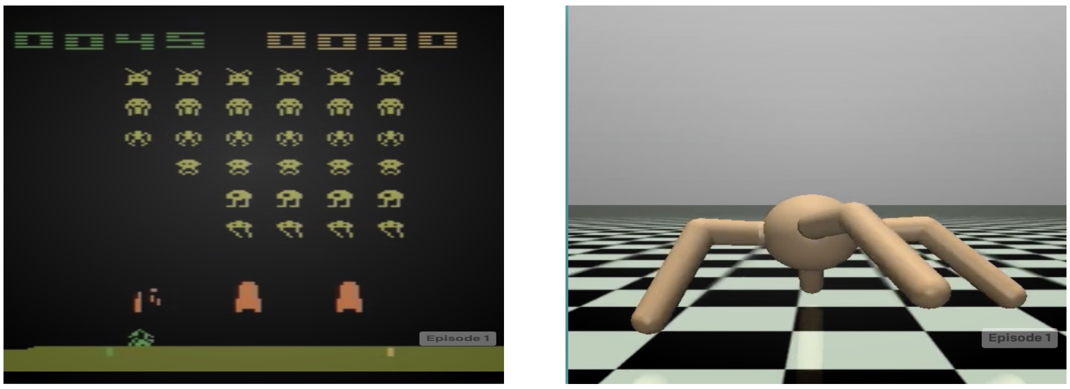


Figure 4.1: Examples of OpenAI neural networks

In conclusion, algorithms that use reinforcement learning are slow to learn. For these examples, it is true that the AI is able to solve certain problems for the scenario where it has trained, but the hours of training required to develop these mechanics are too many. Also, a change in the practice environment would require new training, which would increase the cost of obtaining a self-sufficient AI.

4.1.2 Face detection with OpenCV

A very common problem that artificial intelligence has had to face thousands of times is face detection, and in its improved version, facial recognition. It is already normal to see how social networks such as Twitter, Facebook or Instagram use algorithms to enlarge parts of the image that seem important, or on the other hand, how cell phones are unlocked with the biometric data of our faces.

All these functions that seem so everyday nowadays come from the development of deep learning techniques in image processing.

Therefore, in order to take a small step further, for this section we have used the OpenCV public library, following a guided article [24], which is already trained and whose access is quite simple with a few lines of code.

```
import cv2
import os


def face_detection(img_path, img_name):
    face_cascade = cv2.CascadeClassifier('face_detector.xml')
    img = cv2.imread(img_path)
    faces = face_cascade.detectMultiScale(img, 1.1, 4)
    for (x, y, w, h) in faces:
        cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

    cv2.imwrite(os.path.join("ImageResults", img_name), img)
    print('Successfully␣saved')
```

The work to be done by the user is to gather a battery of images in order to introduce them as data input to the network. These images and the result after processing them with the network can be seen in Figure 4.2, Figure 4.3, Figure 4.3.

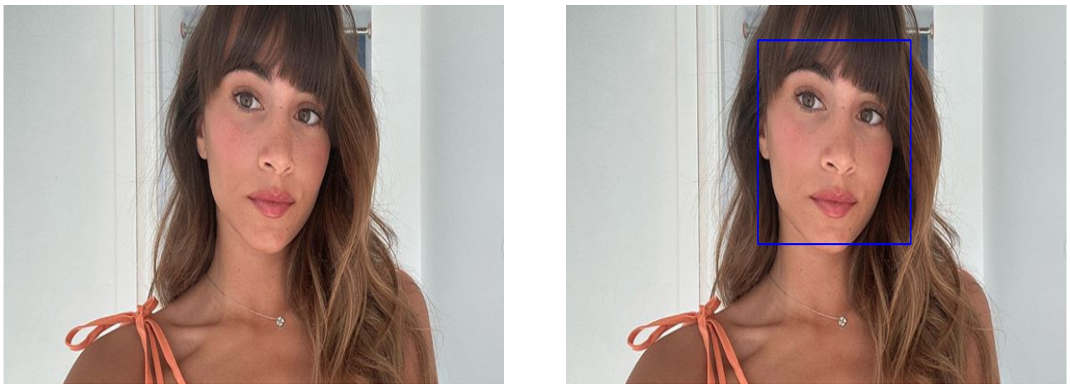


Figure 4.2: Test and result of image 01 comparation

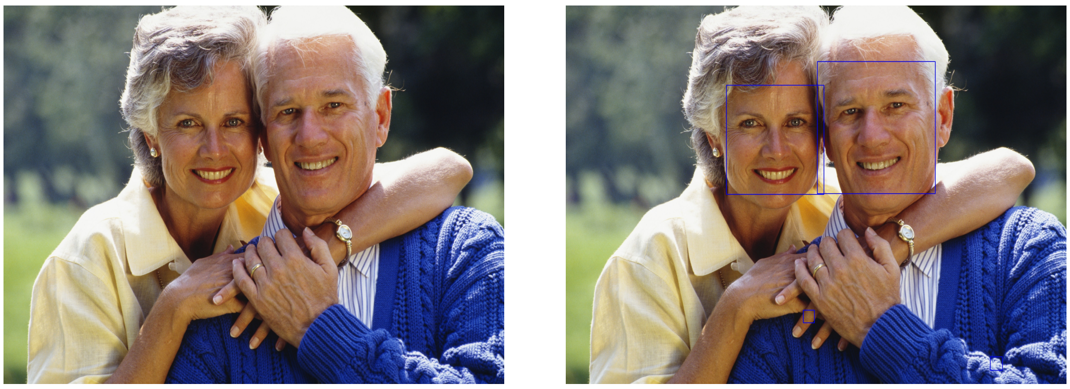


Figure 4.3: Test and result of image 02 comparation

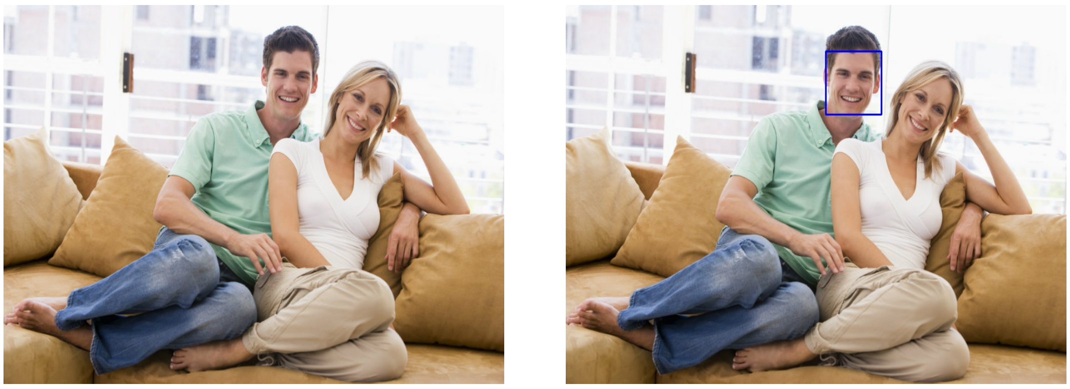


Figure 4.4: Test and result of image 03 comparation

4.1.3 Face aging

As in the previous example, the main function for testing this network was to group a set of images and introduce them as input to the neural model.

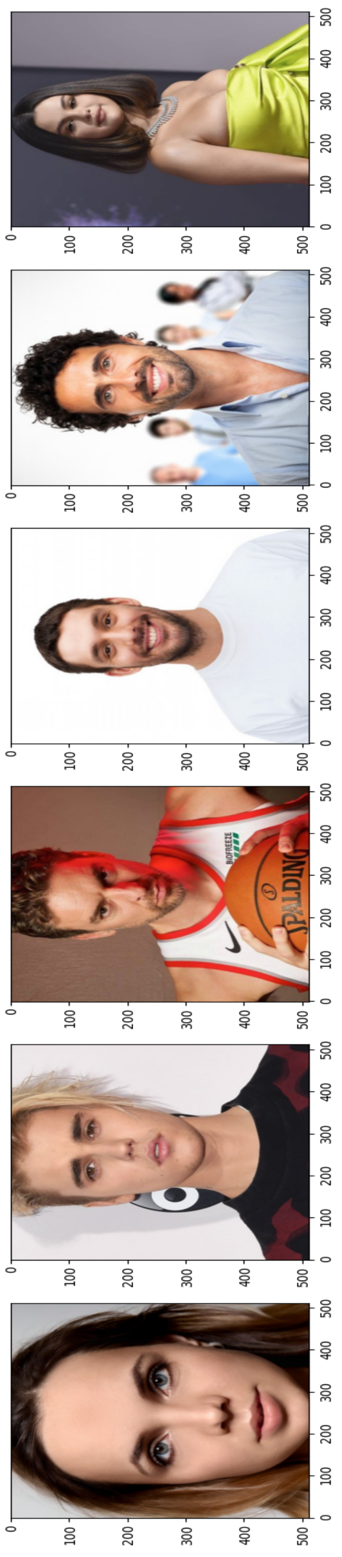
As in the previous example, the code has been modified to read the images through code and not go one by one to see the results of the same.

This has been achieved in a very simple way using the Python OS library and two simple lines of code:

```
        mypath = "ImageTest/"
        onlyfiles = [f for f in listdir(mypath) if isfile(join(mypath, f))]
```

After running the model, the results obtained in the first instance were not very favorable, and it was believed that this was due to the lack of training of the network.

Therefore, unlike the previous example, in this example we have altered the original code to provide a step increase in the training phase and let the network readjust its parameters for three hours to achieve the results shown in Figure 4.5.


(a) Face Aging original images

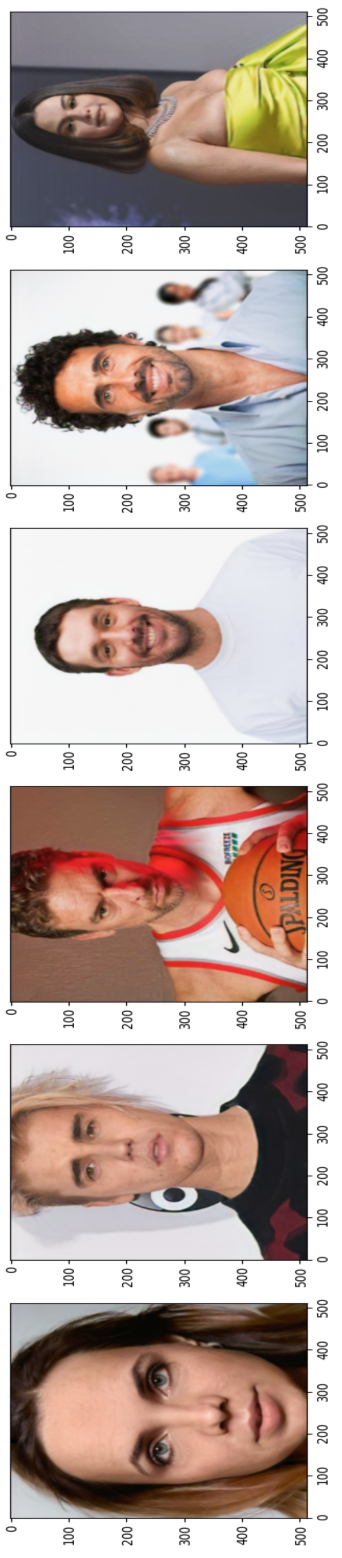

(b) Face Aging processed images' results

Figure 4.5: FaceAging comparison between input and output images

4.2 Guided development, training and testing of neural models

After seeing some real applications without actually programming any of them, just preparing the data to be introduced, the next step of the project was to develop some neural models in a guided way through tutorials.

For this, all the neural networks from now on have been developed and trained with an i9-10900K processor and an NVIDIA 1060 graphics card.

4.2.1 Image resolution upgrade

The next step to the introduction of neural network programming has been to develop an Autoencoder in a guided way in a course taught by Emilio Sansano and his GitHub repository [34]. This model is developed to execute a resolution increase of unsharp or otherwise blurred images.

During the development of this step, it have been learned the main characteristics that define each neural network model and how to define the different layers that will be part of it.

First, the main characteristics that compose a neural network are:

Learning rate: maximum value that can be set for each of the weights of the model.

Sample: each of the different data in the dataset.

Batch size: amount of data in each division of the dataset.

Iteration: number of fractions obtained from dividing the dataset / batchsize.

Epoch: each of the complete cycles that the network has trained using the complete dataset set. In other words, 1 epoch is defined as dataset/batchsize * iterations.

Secondly, to declare the different layers of the network and define the above parameters to be used by the model, the code scheme to be followed is as follows:

```
model = Sequential()
model.add(TypeOfLayer(input_shape), filters,activation)
optimizer = keras.optimizers.Adam(learning_rate)
loss = keras.losses.TypeOfDesiredError
autoencoder.compile(optimizer=optimizer, loss=loss)
history = autoencoder.fit(x_tr_lo, y_tr_hi, epochs=n_epochs
        , batch_size=batch_size, shuffle=True)
```

Thus, the neural model for this application has been schematized in Figure 4.6, showing the different layers that form the network.

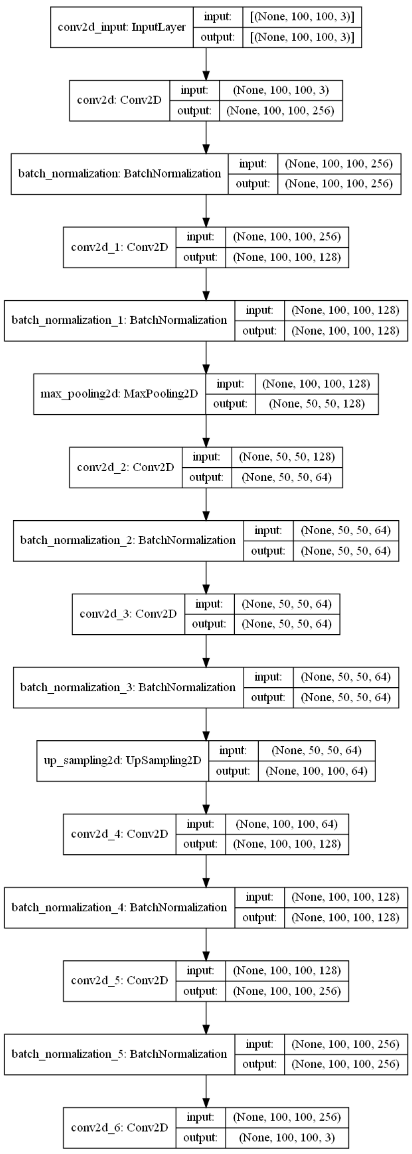

conv2d_input: InputLayer
input: [(None, 100, 100, 3)]
output: [(None, 100, 100, 3)]
conv2d: Conv2D
input: (None, 100, 100, 3)
output: (None, 100, 100, 256)
batch_normalization: BatchNormalization
input: (None, 100, 100, 256)
output: (None, 100, 100, 256)
conv2d_1: Conv2D
input: (None, 100, 100, 256)
output: (None, 100, 100, 128)
batch_normalization_1: BatchNormalization
input: (None, 100, 100, 128)
output: (None, 100, 100, 128)
max_pooling2d: MaxPooling2D
input: (None, 100, 100, 128)
output: (None, 50, 50, 128)
conv2d_2: Conv2D
input: (None, 50, 50, 128)
output: (None, 50, 50, 64)
batch_normalization_2: BatchNormalization
input: (None, 50, 50, 64)
output: (None, 50, 50, 64)
conv2d_3: Conv2D
input: (None, 50, 50, 64)
output: (None, 50, 50, 64)
batch_normalization_3: BatchNormalization
input: (None, 50, 50, 64)
output: (None, 50, 50, 64)
up_sampling2d: UpSampling2D
input: (None, 50, 50, 64)
output: (None, 100, 100, 64)
conv2d_4: Conv2D
input: (None, 100, 100, 64)
output: (None, 100, 100, 128)
batch_normalization_4: BatchNormalization
input: (None, 100, 100, 128)
output: (None, 100, 100, 128)
conv2d_5: Conv2D
input: (None, 100, 100, 128)
output: (None, 100, 100, 256)
batch_normalization_5: BatchNormalization
input: (None, 100, 100, 256)
output: (None, 100, 100, 256)
conv2d_6: Conv2D
input: (None, 100, 100, 256)
output: (None, 100, 100, 3)

Figure 4.6: Model summary of autoencoder

In order to train the network, given a dataset, which in this case is a giant battery of images prepared for the example in question (a random example of such a dataset can be seen in Figure 4.7).

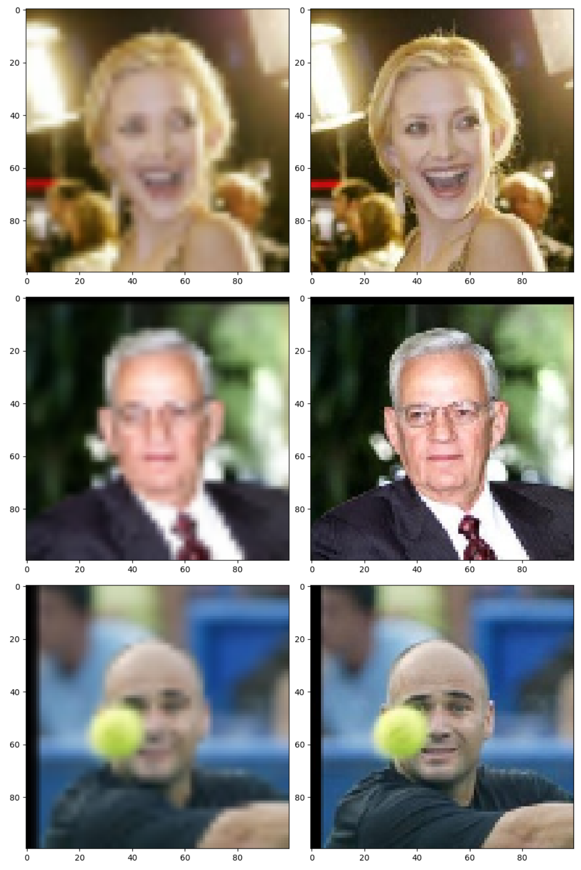


Figure 4.7: Example of one test image introduce to the neural model

Starting from Figure 4.7, on the left is the input image and on the right the expected result. In this way, the network is able to adjust the parameters of the different layers that compose it in order to encode the first image and when decoding it to obtain values similar to those of the second one.

As a result of the training and validation stage, the network generates some error values that can be seen reflected in Figure 4.8.

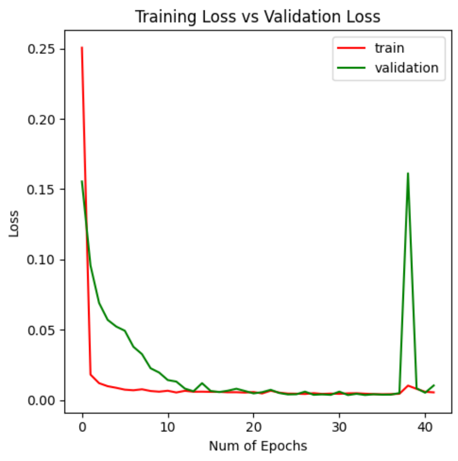


Figure 4.8: Loss value in model history

Finally, from the data set, a random image is chosen and a test is performed. The actual results of the network are shown in Figure 4.9 comparing the input image on the left and the output on the right.

Thanks to the development of this stage, it has been learned to capture the historical data of the model in graphs to provide a visualization of what happens throughout each epoch.

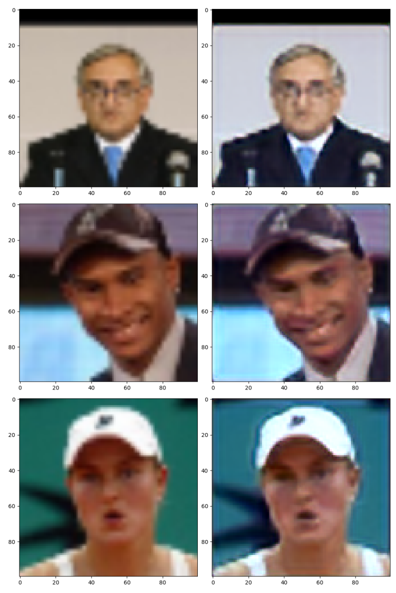


Figure 4.9: Autoencoder result after train the model

4.2.2 Cats and dogs

A very frequent problem when entering the world of neural networks is the classification of a group of images depending on their characteristics [12].

A simple example of what this type of algorithm can be used for can be the calculation of the percentage of women and men who attend a convention through the images obtained in the security camera at the entrance. To do this, the network would have to be trained to distinguish between women and men.

However, in this project, it has been decided to resolve a classification between dogs and cats [39]. Through a CNN, it has been possible to train a neural network to distinguish, through an image given using a graphical interface, if the image in question is a dog or a cat. A CNN model generally consists of convolutional and pooling layers. It works better for data that are represented as grid structures, this is the reason why CNN works well for image classification problems. The dropout layer is used to deactivate some of the neurons and while training, it reduces the offer fitting of the model. To schematize how the network works, in the figure 4.10 it can be seen how it is internally composed, thus defining the type of layer, its inputs, and outputs.

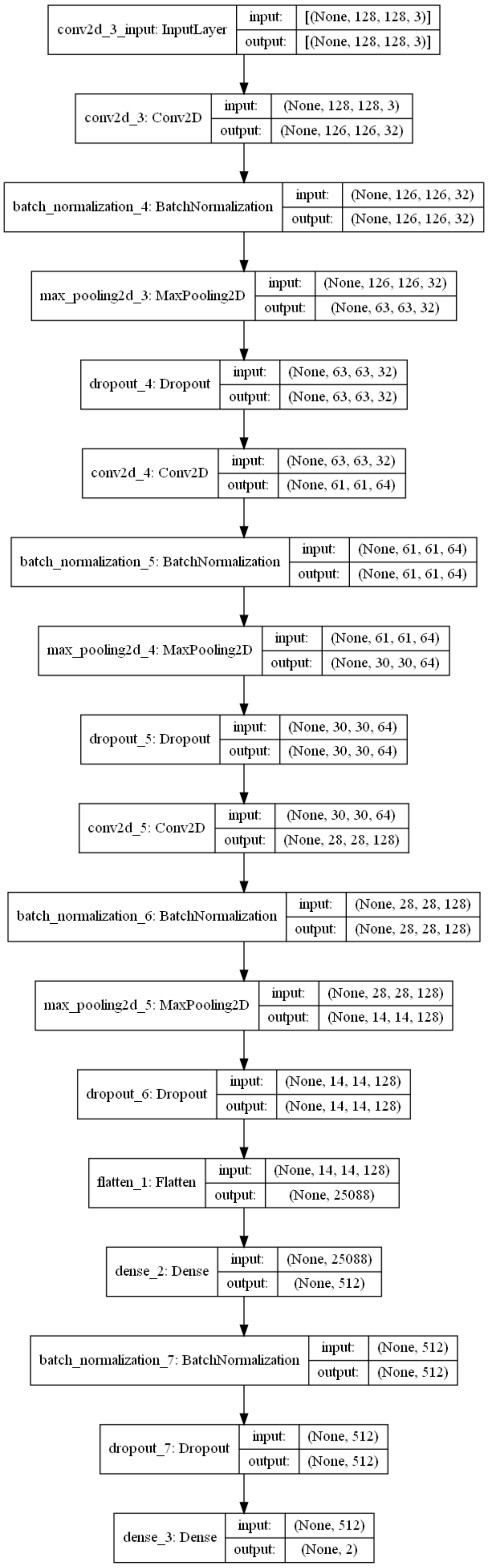

conv2d_3_input: InputLayer
input: [(None, 128, 128, 3)]
output: [(None, 128, 128, 3)]
conv2d_3: Conv2D
input: (None, 128, 128, 3)
output: (None, 126, 126, 32)
batch_normalization_4: BatchNormalization
input: (None, 126, 126, 32)
output: (None, 126, 126, 32)
max_pooling2d_3: MaxPooling2D
input: (None, 126, 126, 32)
output: (None, 63, 63, 32)
dropout_4: Dropout
input: (None, 63, 63, 32)
output: (None, 63, 63, 32)
conv2d_4: Conv2D
input: (None, 63, 63, 32)
output: (None, 61, 61, 64)
batch_normalization_5: BatchNormalization
input: (None, 61, 61, 64)
output: (None, 61, 61, 64)
max_pooling2d_4: MaxPooling2D
input: (None, 61, 61, 64)
output: (None, 30, 30, 64)
dropout_5: Dropout
input: (None, 30, 30, 64)
output: (None, 30, 30, 64)
conv2d_5: Conv2D
input: (None, 30, 30, 64)
output: (None, 28, 28, 128)
batch_normalization_6: BatchNormalization
input: (None, 28, 28, 128)
output: (None, 28, 28, 128)
max_pooling2d_5: MaxPooling2D
input: (None, 28, 28, 128)
output: (None, 14, 14, 128)
dropout_6: Dropout
input: (None, 14, 14, 128)
output: (None, 14, 14, 128)
flatten_1: Flatten
input: (None, 14, 14, 128)
output: (None, 25088)
dense_2: Dense
input: (None, 25088)
output: (None, 512)
batch_normalization_7: BatchNormalization
input: (None, 512)
output: (None, 512)
dropout_7: Dropout
input: (None, 512)
output: (None, 512)
dense_3: Dense
input: (None, 512)
output: (None, 2)

Figure 4.10: Cats and dogs model summary

Once again, the model is trained and as a result, a loss value is generated throughout the training which variations along the time are represented in the figure 4.11

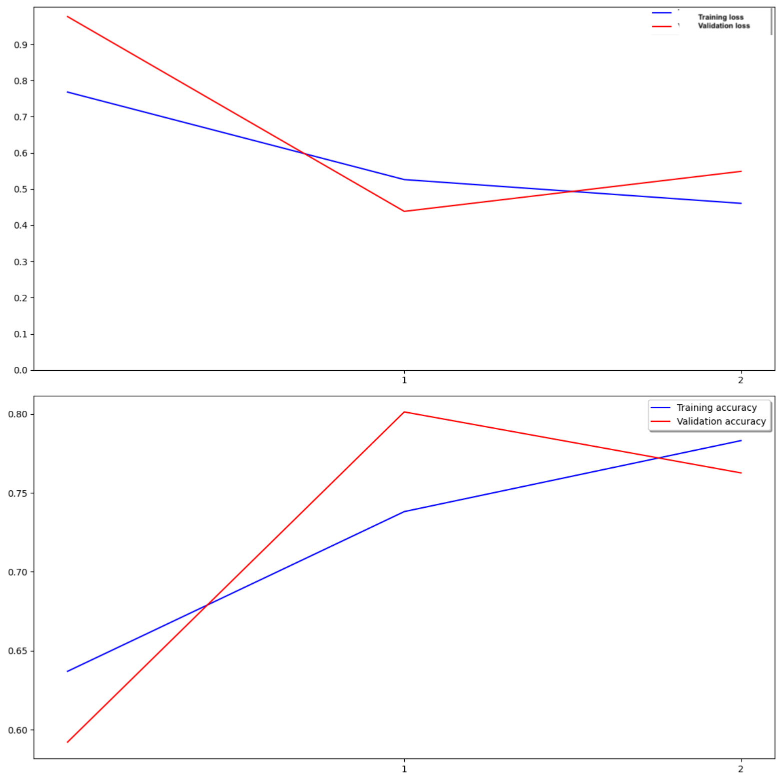


Figure 4.11: Training loss and accuracy vs Validation loss and accuracy along the model history

After the training is complete, the model can classify between dogs and cats with fairly high precision in terms of evaluation. The results can be seen below in figure 4.12

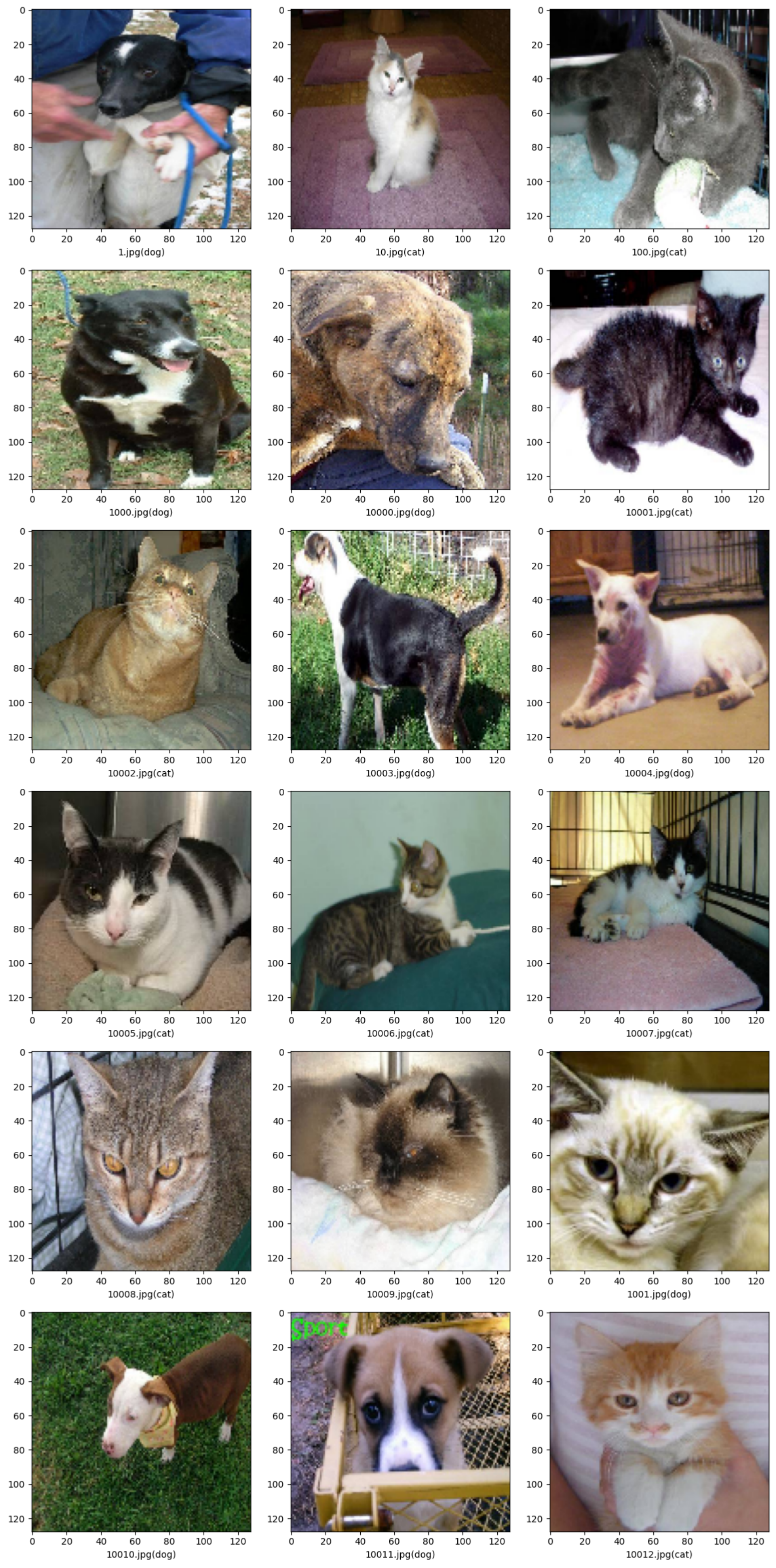


Figure 4.12: A sample of cats and dogs model test data

Finally, a graphical interface has been developed as part of the work in this TFG for the user to upload an image and the trained model to classify it. Thanks to this, it has been learned to develop simple graphical interfaces that can be checked in figure 4.13

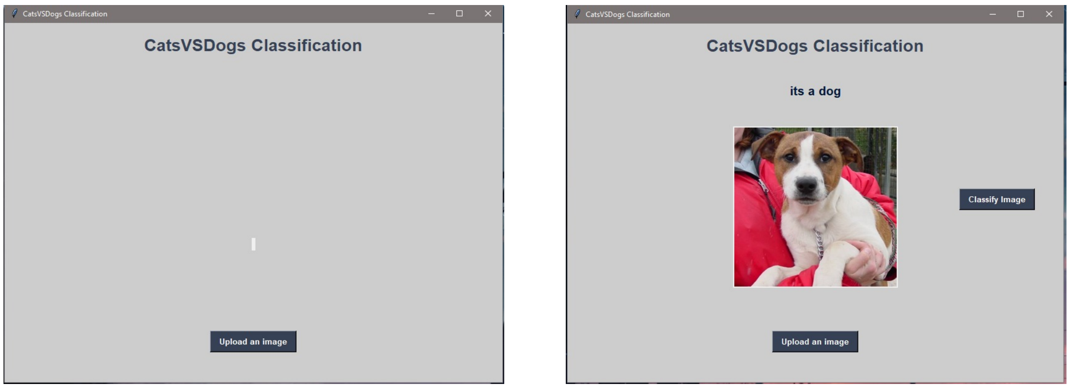

CatsVSDogs Classification
CatsVSDogs Classification
Upload an image
CatsVSDogs Classification
CatsVSDogs Classification
its a dog
Classify Image
Upload an image

Figure 4.13: Python GUI to upload an image to the model and predict if the imagine is either a cat or a dog

4.2.3 Number recognition

Continuing with the development of neural models that can implement a graphical interface, we have proceeded to develop a model capable of reading, interpreting, and processing the reading of handwritten digits [13].

As explained before, is going to be used a CNN model due to its facilities for image data processing. The network layers have been configured as 4.14

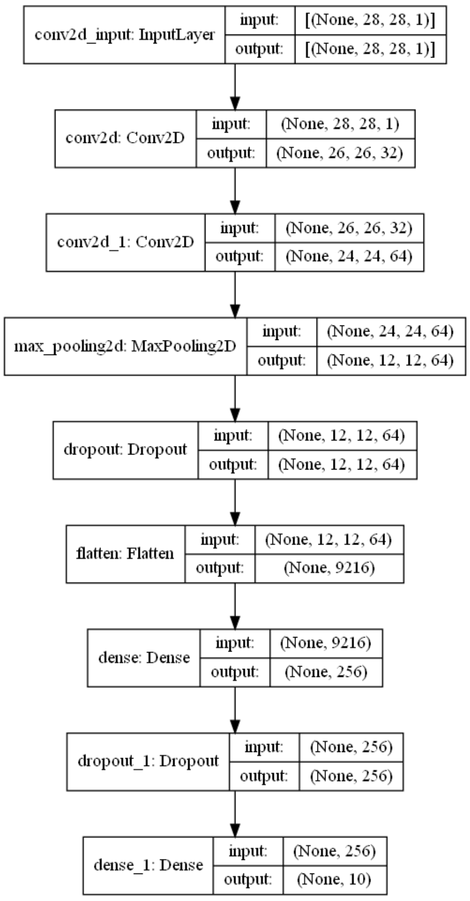


Figure 4.14: Number recognition model summary

The results of the network training have been quite successful and together with the developed interface, it is easy to see which number is predicted by reading the image obtained after typing with the mouse and the probability that the result is correct in figure 4.15.

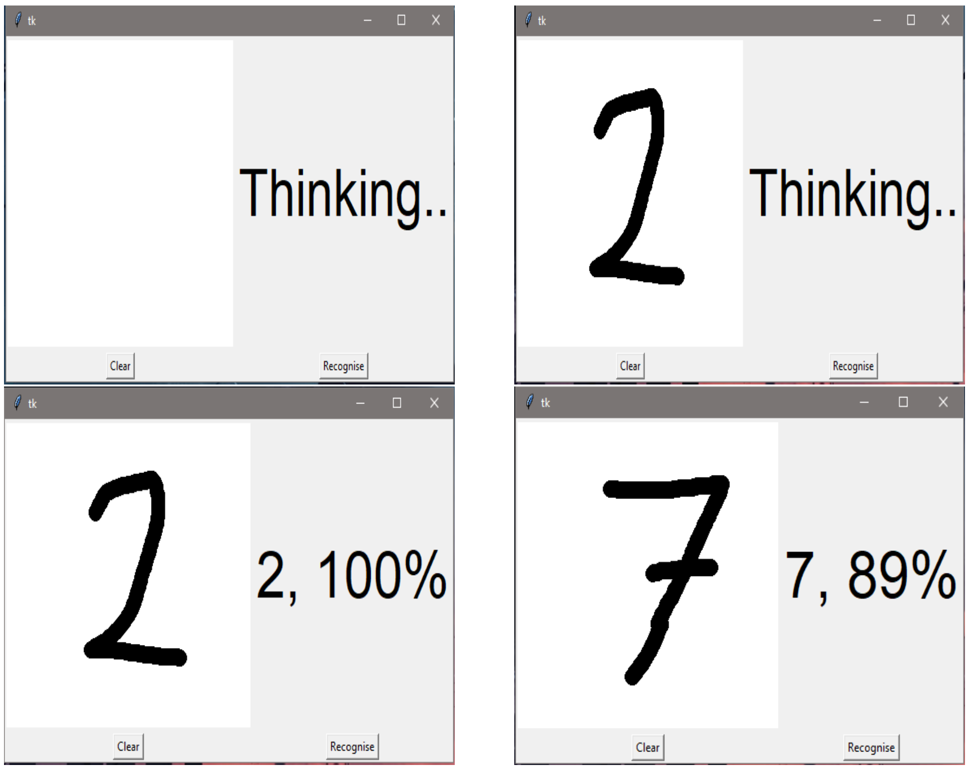


Figure 4.15: Python GUI to introduce handwritten digit and predict its value

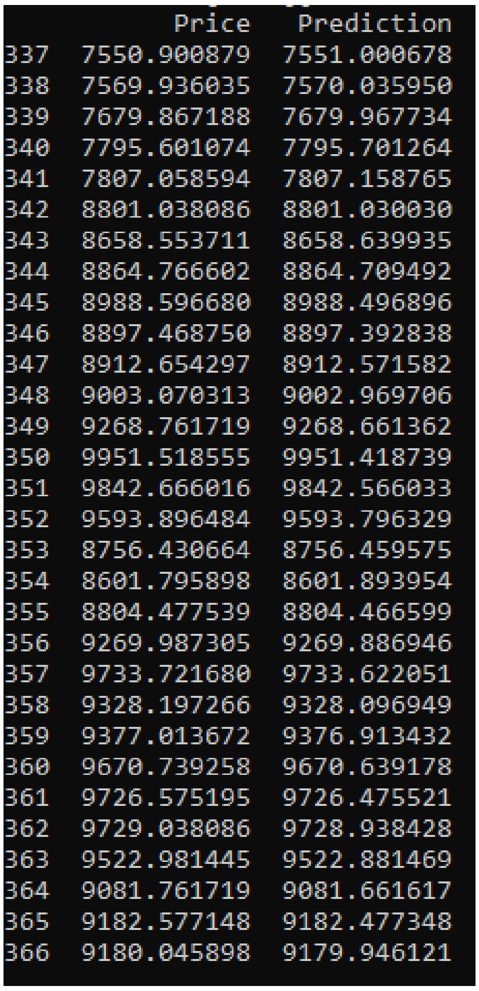
4.2.4 Bitcoin value prediction

As a final step before unguided development, it has been decided to develop an alternative data prediction technique [23] to conventional neural networks, the support vector machines [38] [8].

To do this, the price of Bitcoin is analyzed and its new value is predicted based on its past values, because due to the COVID19 pandemic, the population has been separated into two types of people: those who have started playing paddle tennis and those who have invested in Bitcoin, and it has been seen as an opportunity to apply data prediction techniques to the real world.

The development of an SVM is quite similar to that of a neural network, but unlike the training of neural models, the training and output of this technique has been in a matter of minutes.

The predicted results are quite close to the original results that the model was unaware of at the time of estimation as can be seen in figure 4.16



```
           Price   Prediction
337  7550.900879  7551.000678
338  7569.936035  7570.035950
339  7679.867188  7679.967734
340  7795.601074  7795.701264
341  7807.058594  7807.158765
342  8801.038086  8801.030030
343  8658.553711  8658.639935
344  8864.766602  8864.709492
345  8988.596680  8988.496896
346  8897.468750  8897.392838
347  8912.654297  8912.571582
348  9003.070313  9002.969706
349  9268.761719  9268.661362
350  9951.518555  9951.418739
351  9842.666016  9842.566033
352  9593.896484  9593.796329
353  8756.430664  8756.459575
354  8601.795898  8601.893954
355  8804.477539  8804.466599
356  9269.987305  9269.886946
357  9733.721680  9733.622051
358  9328.197266  9328.096949
359  9377.013672  9376.913432
360  9670.739258  9670.639178
361  9726.575195  9726.475521
362  9729.038086  9728.938428
363  9522.981445  9522.881469
364  9081.761719  9081.661617
365  9182.577148  9182.477348
366  9180.045898  9179.946121
```

Figure 4.16: BTC real values against prediction values

4.3 Application of a neural network to a videogames related problem: matchmaking

As the final step of the project, the aim was to tackle from scratch a real problem related to video games.

To explain the development of this step, the first thing to do is to define the concept of matchmaking.

Matchmaking is a system that tries to match players or teams of players of a similar level or skill by analyzing their statistics from previous games to establish the rank of each player [15].

There are different ways of dealing with this problem, but for the development of a neural network that facilitates the use of this technique, the prediction and estimation of the duration of games has been used as a solution.
How does this solution work?
By analyzing the duration of thousands of multiplayer games, we take into account, for each match, the duration of the match, the ranking of the players in the match, and finally, whether the players are on a winning streak or not. Thus, by averaging the value of all the durations, it can be assumed that a game has been correctly balanced if the duration of the game is greater than the average duration minus the margin of error obtained with the results of a neural network, and that is where the development of the neural network comes in.

In summary, the objective of this section is to develop a neural model capable of predicting the duration of a game given a vector containing the previously mentioned data: ranking of the players in the game and their winning streak.

From the long list of multiplayer video games, it has been decided to use data from public matches of League of Legends [31], a 5vs5 MOBA video game developed by the company RiotGames [32]. This decision has been taken due to the popularity of the game today in e-sports and the familiarity with the game.
With this said, the software development of this project is explained.

First of all, in order to access the public League of Legends games, it is necessary to have a game ID for each game to be analyzed. Once the ID is obtained, the next step is to access the Riot Games database by means of calls through its API [16]. The use [19] of this API is free, but it has limitations, which has been a problem throughout the evolution of the project. The number of calls per second is limited to 2, and every 24h it is necessary to manually regenerate a Key that allows access to the database. In addition to all this, every so often, the server denies access to the database as a security protocol, so it is necessary to make sure that the program that allows downloading the data does not give an error. In this case, the time taken to download data was about 40 hours.

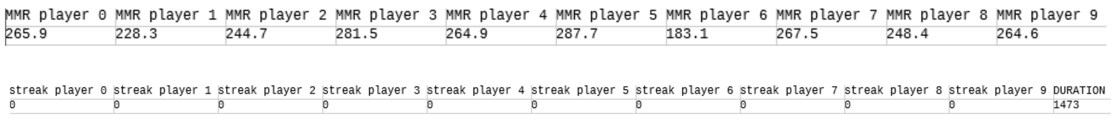
It should be noted that there is a list of IDs published by the Riot developers [17] themselves in order to provide this information. This list has been used for this project.

On the other hand, for each call to the API to obtain the information of the game, 10 more have to be made to obtain the data of each player that forms it. In short, there are many calls, and in order not to have to repeat them, each time all the desired data of a game are obtained, they are stored in a .csv file.

In total, 6209 .csv files have been generated, which means a total of 6209 games where we have stored the ranking of each player that formed it, if they are on a winning streak and the total duration of the game.

It is necessary to emphasize how the calculation of the skill level has been made because given the own ranking of the videogame that is divided in different leagues (bronze, silver, diamond, etc.) composed by 4 divisions and in each division the player can have between 0 and 100 points, it has been assigned in an arbitrary way a representative value of this ranking as the final skill level.

With this already generated, the next step is to generate the neural model to work with. Firstly, a single table has been generated with all the data grouped together as shown in Figure 4.17 and secondly, they have been normalized between 0 and 1 in order to facilitate data processing by the neural network.



MMR player 0	MMR player 1	MMR player 2	MMR player 3	MMR player 4	MMR player 5	MMR player 6	MMR player 7	MMR player 8	MMR player 9
265.9	228.3	244.7	281.5	264.9	287.7	183.1	267.5	248.4	264.6

streak player 0	streak player 1	streak player 2	streak player 3	streak player 4	streak player 5	streak player 6	streak player 7	streak player 8	streak player 9	DURATION
0	0	0	0	0	0	0	0	0	0	1473

Figure 4.17: Processed RiotGames Data before values normalization

Secondly, the neural model is formed by specifying its input data and the layers that form it. The structure of the network has been schematized in Figure 4.18.
With the data obtained, it has been divided into 80% for the training set and 20% for the test and validation set. After that, if a vector of 20 values is entered, it is the network itself that estimates the duration of the game and the user who assesses whether a good match has been made or not.

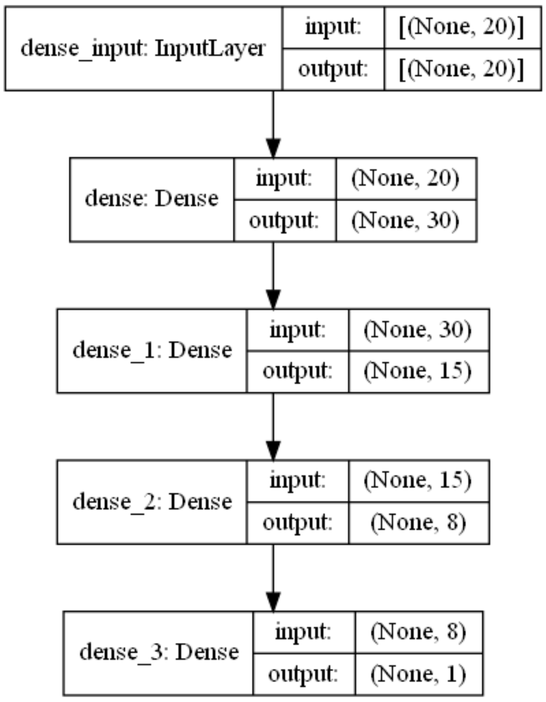


Figure 4.18: Model Summary

As shown in Figure 4.18 the network is composed of different density layers where for each of them a different activation function is applied. As an alternative, we tried to apply a LSTM model [4] but the results were not as expected.

As a summary of the generated network, the loss values generated by the network in the training and test sets are shown in Figure 4.19.

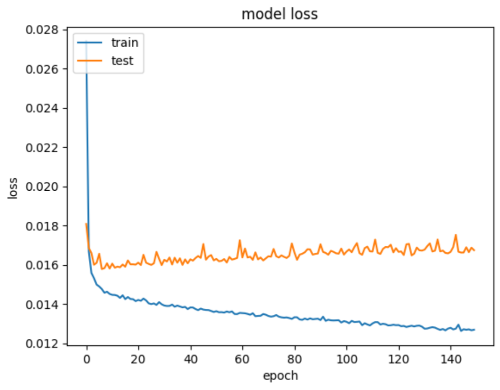


Figure 4.19: Trainning and test history

In addition, in order to see the different error values produced by the network throughout its training stages, the different types of errors throughout this stage are shown in Figure 4.20.

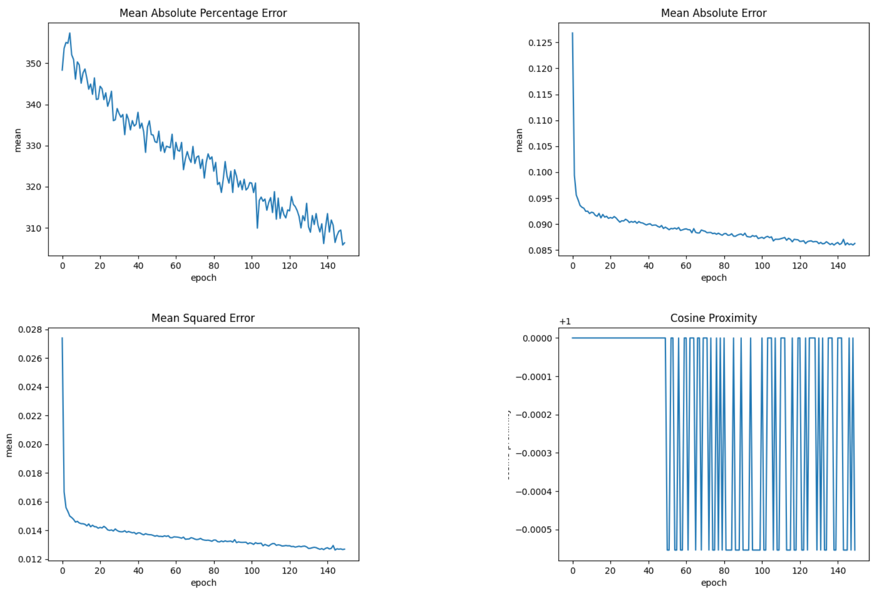
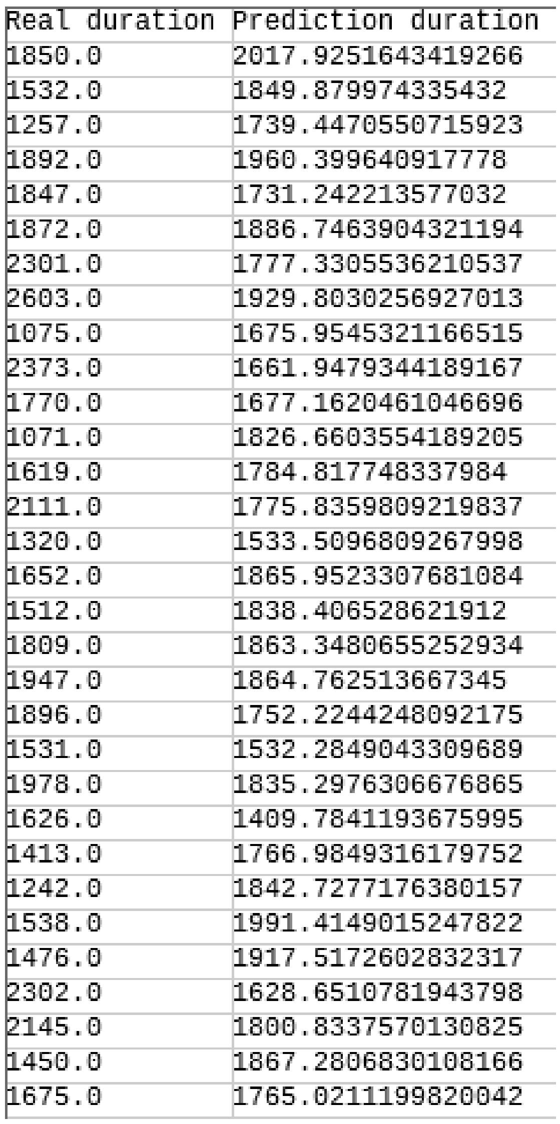


Figure 4.20: Error values in trained model

After two hours of training the network, the values produced in the validation test can be seen compared in Figure 4.21 where on the right are the estimated values and on the left the real values of the game.



Real duration	Prediction duration
1850.0	2017.9251643419266
1532.0	1849.879974335432
1257.0	1739.4470550715923
1892.0	1960.399640917778
1847.0	1731.242213577032
1872.0	1886.7463904321194
2301.0	1777.3305536210537
2603.0	1929.8030256927013
1075.0	1675.9545321166515
2373.0	1661.9479344189167
1770.0	1677.1620461046696
1071.0	1826.6603554189205
1619.0	1784.817748337984
2111.0	1775.8359809219837
1320.0	1533.5096809267998
1652.0	1865.9523307681084
1512.0	1838.406528621912
1809.0	1863.3480655252934
1947.0	1864.762513667345
1896.0	1752.2244248092175
1531.0	1532.2849043309689
1978.0	1835.2976306676865
1626.0	1409.7841193675995
1413.0	1766.9849316179752
1242.0	1842.7277176380157
1538.0	1991.4149015247822
1476.0	1917.5172602832317
2302.0	1628.6510781943798
2145.0	1800.8337570130825
1450.0	1867.2806830108166
1675.0	1765.0211199820042

Figure 4.21: Real values vs estimation values

Calculating the estimation error for each element, an average error of ± 147.20 was obtained. Thanks to this error, it can now be assumed that: given 10 players along with the information whether they are on a winning streak or not, a good match will be accepted if

$$GamePredictionResult \geq MeanTimeOfAnalyzedGames - MeanError$$

thus having a possible solution to the matchmaking problem in 5vs5 video games.

4.4 Project development writing and presentation of the project

An important part of this project is the drafting of the project report. For this reason, it is important to highlight the time spent on this task, since we had to learn a new writing language: latex.

In addition, in order to obtain a document written in formal English, two different tools were used: DeepL as a Spanish - English translator, and Grammarly as an application of cohesion and coherence of the text in English.

In order to obtain the best result, each section of this project has been written and translated using DeepL and in order to obtain a good result, after processing each paragraph by Grammarly, if a score of more than 85/100 was not obtained, the written paragraph was restructured in order to improve the text.

On the other hand, for the presentation of the text, a previous preparation is foreseen, which includes different tests and the formalization of the visual presentation to be used.

Chapter 5

Results and objectives

Contents

5.1 Results

As a final part of the project development, it is necessary to analyze the results of the project taking into account what has been learned and how I have dealt with the different complications that have arisen throughout the project development process.

So, the question to answer is, what have I achieved with this project?

- I have learned what a neural network is and how they work internally so that I have been able to develop one from scratch.
- I have been able to access a real database using public API calls.
- I have learned how to use Tensorflow for the management and use of neural networks and complementary libraries in order to display the data produced during the training of a neural network in graphs.
- I have developed graphical interfaces with Python so that the standard user has greater accessibility without the need to understand how the terminal of his operating system works.

- I have been able to create readable tables of data using the Pandas library and creating .csv extension files.
- I have parsed json files from code and I have extracted the data that I needed at all times

In short, I have learned more than I imagined thanks to this project. From the initial concepts that I proposed to myself to the problems that I have been facing throughout the development of the project.

Thus, it is time to make a comparison between the objectives of the project and the achievements.

5.2 Objectives

The main objectives of the project can be listed as:

- To understand in general terms how primary neural networks work ✓
- To puzzle out which are the different existing deep learning techniques and what are they used to ✓
- To study how problems related to video games can apply these techniques ✓
- To learn to use the meaning techniques of deep learning using public libraries ✓
- To design a problem related to video games and provide a solution to it by applying what has been learned in the previous steps ✓

All the objectives that were proposed in the beginning, have been achieved, so it can be said that the project has been a success.

5.3 Access to the project

For anyone who wants to test the project or wants to see how it has been developed, here below is a link to a GitHub repository where you can find all the necessary tools to understand and test it.

https://github.com/victor13alvarez/FinalDegreeProject_MachineLearning

CHAPTER 6

Conclusions and Future Work

Contents

In this chapter, the conclusions of the work, as well as its future extensions are shown.

6.1 Conclusions

Overall, the work experience has been very good. At the beginning of the work, I knew absolutely nothing about neural networks and the countless applications of trained models. However, today I not only know the main deep learning techniques, but I was able to develop my model by accessing a public database using a public API.

It is a pity that after four years of career, artificial intelligence is nothing more than a subject related to pathfinding in enemy movement or simulation of decision trees in terms of strategic behavior.

That is why I think that the decision to carry out this research project has been a wise decision, because AI techniques are becoming more and more advanced and having a base to start working with is a great step to start working on this project.

6.2 Future work

As for the future of the project, I think that it is impossible to know everything about a subject, therefore it is impossible to say that the project has finished for me.

If I focus only on the last point of this project: estimating the duration of games to predict if the matchmaking has been reasonable or not, I think that probably, with more experience and adjustment of parameters, more realistic results can be reached. And if I think about the entire project, I would like to learn a lot more about deep learning techniques and artificial intelligence in general.

Today, my plan for the future is to learn about audio processing through deep learning techniques to process my voice and create a virtual assistant capable of imitating the main functions of the most popular virtual assistants such as Siri, Cortana o Alexa to enhance its use in the field of mobile applications or video games.

On the other hand, I would like to develop my API to see the other side of how APIs work because I have learned what they are and how to use them, but it is time to learn how to develop them.

This project has opened my mind to start many other ideas related to the main theme of the project: Deep Learning Techniques Applied to Videogames.

Bibliography

[1] Open AI. Open ai gym. https://gym.openai.com.

[2] Redacción APD. Deep learning vs machine learning: qué las diferencia. https://www.apd.es/deep-learning-vs-machine-learning-las-diferencia/.

[3] Avijeet Biswal. Top 10 deep learning algorithms you should know in 2021. https://www.simplilearn.com/tutorials/deep-learning-tutorial/deep-learning-algorithm.

[4] Jason Brownlee. How to make predictions with long short-term memory models in keras. https://machinelearningmastery.com/make-predictions-long-short-term-memory-models-keras/.

[5] James Chen. Neural network. https://www.investopedia.com/terms/n/neuralnetwork.asp.

[6] Hsu-Wen Chiang. https://slidetodoc.com/status-report-on-machine-learning-hsuwen-chiang-le/.

[7] Dot CSV. ¿qué son las redes neuronales? https://www.youtube.com/playlist?list=PL-Ogd76BhmcB9OjPucsnc2-piEE96jJDQ.

[8] Gabriele de Luca. Neural networks vs support vector machines: are the second definitely superior? https://www.baeldung.com/cs/svm-vs-neural-network.

[9] DeepMind. Alpha star. https://en.wikipedia.org/wiki/AlphaStar_(software).

[10] DeepMind. Deep mind. https://deepmind.com.

[11] EndtoEnd.ai. Atari environments. https://www.endtoend.ai/envs/gym/atari/.

[12] Data flair. Cats and dogs classification. https://data-flair.training/blogs/cats-dogs-classification-deep-learning-project-beginners/.

[13] Data flair. Handwritten number recognition. https://data-flair.training/blogs/python-deep-learning-project-handwritten-digit-recognition/.

[14] Jake Frankenfield. Artificial intelligence. https://www.investopedia.com/terms/a/artificial-intelligence-ai.asp.

[15] Diccionario online de términos sobre videojuegos y cultura gamer GamerDic. Definición de matchmaking. http://www.gamerdic.es/termino/matchmaking.

[16] Riot games. Riot api documentation. https://developer.riotgames.com/.

[17] Riot Games. Riot developer portal. https://developer.riotgames.com.

[18] Javier García. Redes neuronales - fácil y desde cero. https://www.youtube.com/playlist?list=PLAnA8FVrBl8AWkZmbswwWiF8a_52dQ3JQ.

[19] Barney H. How to use riot api. https://towardsdatascience.com/how-to-use-riot-api-with-python-b93be82dbbd6.

[20] HasnainRaz. Fastaging with a gann. https://github.com/HasnainRaz/Fast-AgingGAN.

[21] Steffen Hölldobler, Sibylle Möhle, and Anna Tigunova. Lessons learned from alphago. 06 2017.

[22] Will Kenton. Multiple linear regression. https://www.investopedia.com/terms/m/mlr.asp.

[23] Aman Kharwal. Bitcoin price prediction. https://thecleverprogrammer.com/2020/05/23/bitcoin-price-prediction-with-machine-learning/.

[24] Aman Kharwal. Face detection with python. https://thecleverprogrammer.com/2020/10/09/face-detection-with-python/.

[25] Jordi Mansanet. Machine learning y casos de uso. http://decharlas.uji.es/es/introduccion-machine-learning.

[26] mgbellemare. Arcade learning environment. https://github.com/mgbellemare/Arcade-Learning-Environment.

[27] Christopher Olah. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/#lstm-networks.

[28] OpenAI. https://openai.com.

[29] OpenAI. https://openai.com/projects/five/.

[30] OpenAI. https://openai.com/blog/emergent-tool-use/.

[31] RiotGames. League of legends. https://euw.leagueoflegends.com/es-es/how-to-play/.

[32] RiotGames. Riotgames. https://www.riotgames.com/es.

[33] Emilio Sansano. Deep learning: Una introducción práctica. http://decharlas.uji.es/es/introduccion-deep-learning.

[34] Emilio Sensano. Tutorial deep-learning. https://github.com/esansano/tutorial-dl.

[35] Statistics Solutions. What is linear regression? https://www.statisticssolutions.com/what-is-linear-regression/.

[36] CG Trikcs. Nvidia ai playground | gaugan beta. https://www.youtube.com/watch?v=UpmqR2ZCPFQ.

[37] Unknown. 7 types of neural network activation functions: How to choose? https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/.

[38] Unknown. Neural networks vs support vector machines: are the second definitely superior? https://stats.stackexchange.com/questions/30042/neural-networks-vs-support-vector-machines-are-the-second-definitely-superior.

[39] Uysimty. Cats and dogs classify. https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification.

[40] Wikipedia. Alphago. https://en.wikipedia.org/wiki/AlphaGo.

[41] Wikipedia. Deepblue. https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer).

[42] Wikipedia. Go, the game. https://es.wikipedia.org/wiki/Go.

[43] Wikipedia. Minimax. https://es.wikipedia.org/wiki/Minimax.

[44] Wikipedia. Monte carlo tree search. https://en.wikipedia.org/wiki/Monte_Carlo_tree_search.

Appendix A

Other considerations

All the repositories of interest to learn how to use and develop neural networks are included either in bibliography section or below to provide an easy way to access the same information that has been followed along the project development filtered by non-research work links so any user that has the same interest the project was started for can try and test some guided examples that explain every step of neural networks development.

- https://thecleverprogrammer.com/2020/05/23/bitcoin-price-prediction-with-machine-learning/
- https://thecleverprogrammer.com/2020/10/09/face-detection-with-python/
- https://data-flair.training/blogs/
- https://www.kaggle.com/uysimty/keras-cnn-dog-or-cat-classification
- https://github.com/esansano/tutorial-dl

Finally, a special mention to Emilio Sasnsano and his introductory talks on neural networks and deep learning.