



The
University
Of
Sheffield.

Comparison of Methods for Table Tennis Ball Prediction

A thesis submitted in fulfilment of the requirements for the
degree of:

Master of Philosophy

The University of Sheffield

Department of Automatic Control and Systems Engineering

May 2021



Abstract

Recent developments in computer science have led machine learning to become one of the most used tools in artificial intelligence in modern times. It is applied in many areas of practical life and not limited to academia or engineering. This is due to machine learning's flexibility that makes it applied easily in a variety of problems areas, which can be solved if the data is properly managed.

This thesis focuses on using a machine learning approach to a problem that has been addressed in publications many times using model predictive control.

The similarities that will allow mathematical modelling to be replaced with a machine learning approach will be analysed and evaluated and ultimately two approaches will be implemented.

The problem to be solved is to predict a table tennis ball while uncertainties arise in the sensing process in an average game of table tennis.

The results of this research are compared with current different approaches to solve the prediction of the ball. The focus and novelty lie in the improved accuracy of the predictions using suitable neural networks architectures.



Acknowledgement

I wish to express my deepest gratitude to my supervisors Professor Sandor M. Veres and Dr J Anthony Rossiter for their support, help, tolerance and above all their patience during my time at Sheffield University. Their attention and understanding were crucial to the completion of this thesis.

I am grateful to the University of Sheffield for giving me the opportunity to pursue my academic research and training.

My heartfelt thanks go to my mother, brother and friends for their unconditional support throughout my life.

Nomenclature

Mathematical notation

m	Mass
C_D	Coefficient of air resistance
p	Air density
V	Velocity vector
F	Force vector
A	Cross-section of the area of the ball
ω	Ball spin acceleration
C_M	Magnus force coefficient
r	Ball radius
g	Gravity

Neural Networks

x	Neuron inputs	w_{ij}	Convolution filter
ω	Weights	I	Image
y	Neuron activation	U	Pixels
θ	Neuron threshold	c	Colours
P_k	Perceptron weights	I_{Fg}	Background frame
n	Kernel value	I_g	Greyscale
m	Feature map	T	Tolerance



List of figures

2.1 Forpheus at the ces 2018 [90]	6
2.4.1 Example of a typical control method for a table tennis robot [9]	11
3.1.5.1 Graphical example of a typical threshold logic unit	24
3.1.5.2 Graphical example of a step up response	25
3.1.5.3 Graphical example of a sigmoid response	25
3.1.6.1 Graphical example of a perceptron	26
3.1.7.1 Graphical example of a multilayer network	28
4.1.1 Proposed setup consisting of two cameras, a and b, distributed above and on one side of the table respectively	38
4.2.1 Proposed methodology to train the neural network	43
4.2.1.1 Example on defining an area of interest on a screenshot	44
4.2.2.1 Graphical example of the results of different thresholds	46
4.2.2.2 Table segmentation and ball contact area examples	48
4.2.2.3 Graphical representation of the ball being detected on a frame with the grid in red. The yellow segments represent the segments activated due to the ball position in the grid. The red segments represent the segments where the ball have been detected previously	50
4.2.3.1 Graphical representation of the ball trajectory. Result of adding all the frames when the ball was being detected (left), applying the grid to the frame (middle) and flagging the segments where the ball was detected (right)	50
4.5.1 Proposed neural network methodology	57
4.5.2 Proposed genetic programming training methodology	58
5.1.1 Graphical representation of the tree obtained through genetic programming	60
5.1.2 Pareto front graphic comparing the fitness function to the number of nodes applied and their current population	61
5.1.3 Graphic representing the gradient of the fitness function	62
5.1.4 Graphic comparing the accuracy against the complexity of the fitness level	62
5.1.5 Three graphics representing the result of fitting the curve with genetic programming with different amount of generations available, 72 (top), 98 (middle) and 200 (bottom)	64
5.2.1 Graphic representing the accuracy of the convolutional neural network	70
5.2.2 Graphic representing the gradient results of the convolutional neural network	71
5.2.3 Graphic representing the training performance of the network	72



5.2.4 Graphic representing the error histogram of the network 73

5.2.5 Graphic representing the positive errors while classifying the inputs 74 es..... 81





List of tables

3.1.7.1 Table representing the steps to apply the backpropagation method on a hidden layer network.	31
4.1.2.1 Table comparing the main disadvantages observed when implementing different approaches with a simple machine learning exercise on the raspberry pi	41
4.4.1.1 Comparison of weakness and fortitudes of the current methodology to predict the ball final position against the proposed neural network approach	53
5.2.1 Table showing the result of using two neurons in each hidden layer	67
5.2.2 Table showing the result of using five neurons in each hidden layer	68
5.2.3 Table showing the result of using 10 neurons in each hidden layer	69



Content

CHAPTER 1

INTRODUCTION	1
1.1 PROBLEM STATEMENT.....	1
1.2 JUSTIFICATION.....	2
1.3 OBJECTIVES.....	2
<i>1.3.1 General objective</i>	<i>2</i>
<i>1.3.1 Supporting objectives.....</i>	<i>3</i>
1.4 SCOPE.....	3
1.5 CONTRIBUTIONS	3
1.6 THESIS OUTLINE.....	4

CHAPTER 2

LITERATURE REVIEW	5
2.1 CURRENT TABLE TENNIS ROBOTS.....	5
2.2 NONLINEAR MODELLING APPROACH.....	6
2.3 BALL DETECTION AND PREDICTION.....	8
2.4 CONTROL OF THE BALL	10
2.5 MACHINE LEARNING IN TABLE TENNIS	12
<i>2.5.1 Machine learning applied to predicting the ball trajectory</i>	<i>13</i>
<i>2.5.2 Machine learning applied to controlling the ball trajectory</i>	<i>13</i>

CHAPTER 3

TECHNICAL PRELIMINARIES	15
3.1 MACHINE LEARNING	15
<i>3.1.1 Unsupervised learning</i>	<i>16</i>
<i>3.1.2 Supervised learning.....</i>	<i>16</i>
<i>3.1.3 Genetic programming.....</i>	<i>17</i>
<i>3.1.4 Neural networks.....</i>	<i>21</i>
<i>3.1.5 Threshold Logic Unit</i>	<i>23</i>
<i>3.1.6 Perceptron</i>	<i>26</i>
<i>3.1.7 Networks with multiple layers</i>	<i>28</i>
<i>3.1.8 Convolutional neural network.....</i>	<i>32</i>
3.2 MACHINE LEARNING APPROACH COMPARISON.....	35
3.3 OBJECT TRACKING	36

CHAPTER 4

METHODOLOGY	37
--------------------------	-----------

4.1	TABLE TENNIS BALL RECORDING SETUP	37
4.1.1	<i>Raspberry Pi</i>	39
4.1.2	<i>Matlab</i>	40
4.2	DATABASE FOR MACHINE LEARNING	42
4.2.1	<i>Area of interest</i>	43
4.2.2	<i>Camera frames processing</i>	45
4.2.3	<i>Image output</i>	50
4.2.4	<i>Convolutional neural network class</i>	51
4.3	SUITABLE MACHINE LEARNING TOOLS	51
4.3.1	<i>Testing of the selected machine learning tools</i>	52
4.4	MATHEMATICAL MODEL REPLACED WITH GENETIC PROGRAMMING	52
4.4.1	<i>Mathematical model replaced with convolutional neural network</i>	53
4.5	PREDICTION OF THE BALL AND EXPECTED RESULTS	56
 CHAPTER 5		
EXPERIMENTS AND RESULTS		59
5.1	GENETIC PROGRAMMING	59
5.2	CONVOLUTIONAL NEURAL NETWORK	66
 CHAPTER 6		
CONTRIBUTIONS, CONCLUSIONS AND FUTURE WORK		76
6.1	CONTRIBUTIONS	76
6.2	CONCLUSIONS	77
6.3	FUTURE WORK	79
REFERENCES		81



Chapter 1


Introduction

Table tennis robots are currently used in control science as tools to prove the capability of modern predictive techniques to control the trajectory of the ball within the limits of table tennis rules [5], [10], [14], [16], [41], [51], [81]. In order to control the ball, it is necessary to predict the ball trajectory to ensure contact between the bat and the ball.

In most of the related research publications [12], [21], [63], [86], the principles of computational steps that are needed to control the ball are presented similarly, but the methods used to accomplish the steps vary greatly. Due to this variation in methods, the most basic and common approaches are reviewed and explained to reflect the most current research, in order to have a better idea of the reasoning that led to the particular methodology employed in this thesis.

1.1 Problem statement

Current robots capable of playing table tennis are not able to play competitively against a human player due to their very limited control over the ball [33], [62]. This limited control is not due to hardware issues. This is justified on the grounds that the robots can be observed to be capable of achieving suitably fast speeds and degrees of precision [14], [15], [16]. Due however to the inconsistencies in the design of the prediction step, the ball is predicted with errors that cause miscalculation in the angle of the bat for controlling the ball. This can even cause the ball to bounce out of bounds with large errors in the returns [24].



In this research, some of the most regularly observed errors, which cause this problem, will be investigated and a solution will be proposed alongside the reasoning that led to this solution.

1.2 Justification

The ability of a robot to predict and control a moving object is of great importance in several areas of research. While some of the current activities and tasks that robots are able to accomplish in industry are being performed with an ample frame of time [39], [60], some of the most challenging and important problems are limited not only in time but also in resources, such as the physical space available.

These time constraints alongside the physical limits are present in the sport of table tennis. Robots capable of performing correctly in this problem may provide viable solutions to other similar problems [47].

1.3 Objectives

As stated previously, in order to control the table tennis ball, there are several tasks that need to be observed; consequently several objectives need to be achieved.

1.3.1 General objective

Successfully predict the trajectory of a table tennis ball through a machine learning approach which considers the nonlinearity of the ball.



1.3.1 Supporting objectives

- Track the table tennis ball and generate a usable database for a machine learning approach.
- Predict the trajectory of the ball without spin through machine learning.

1.4 Scope

This research aims to prove that a machine learning approach is capable of replacing a traditional prediction of the ball, while retaining a similar degree of accuracy and precision.

1.5 Contributions

The result of this research is a novel method which applies machine learning to solve a very particular problem, which provides an example for future researchers wanting to solve similar problems.

A potential impact of this research could lead to a commercial robot that can carry out training alongside other robots or with humans at a basic and intermediate level of table tennis and steadily improves its play.



1.6 Thesis outline

- Chapter 1: The first chapter contains the basic information regarding the structure of the thesis, the objectives and the hypothesis.
- Chapter 2: The second chapter review the literature that sustains the current research and details the general research approach that has been done regarding the specific problem established in the previous chapter.
- Chapter 3: The third chapter focus on the technical preliminaries of the research, explains in detail how the machine learning tools work and their general implementation.
- Chapter 4: In this chapter the novel methodology is introduced and justified. The reasoning on how to properly address the problem to have a viable solution is argued.
- Chapter 5: The experiments and results are explained in detail. Following the previous chapter's theory, the results are now presented in empirical detail.
- Chapter 6: This chapter summarizes the contributions of this research. Conclusions are drawn and future work is outlined. Some potential future applications of this research are also listed.



Chapter 2

Literature review

This chapter provides a summary of the current state of research regarding table tennis robots and some topics of table tennis. These topics are relevant for the control of the ball and will be reviewed in detail.

2.1 Current table tennis robots

The majority of table tennis robots are designed to solve a particular challenge in terms of the table tennis methodology, so the robots are designed, constructed and tested in specific controlled settings. This does not mean that all the current table tennis robots are used exclusively in laboratories [65] [67] [69], there are a handful of robots that are constantly shown to the public and used in more general settings to test a more realistic approach. An example of this case is the robot FORPHEUS [90], from the OMRON corporation.

While some of the reviewed robots are demonstrated in real scenarios, they still fail to tackle efficiently the problem of predicting the ball when it is spinning.



Figure 2.1 FORPHEUS at the CES 2018 [90].

2.2 Nonlinear modelling approach

The ball spin has been a matter of “debate” for the last few years in terms of table tennis prediction. Some researchers consider it an essential part of the game [1], [2], [3], [4], while other researchers argue that the spin of the ball could be ignored due to the relatively small effect it might have on fast moving balls [5], [6], [7]. This inconsistency on how to properly manage the spin of the ball has caused several researchers to come up with different methods to simplify the spin to keep using the previous mathematical models [2].

Apart from ball aerodynamics, the spin of the ball is the main factor that causes the system to be considered nonlinear. As such, the approach to properly deal with this problem has been the subject of research for the last few years and some researchers attempted to find ways to calculate the spin of the ball in novel ways and realise how exactly this spin is influencing the trajectory of the ball in the air and when it bounces on the table [21], [22], [24], [25], [26], [27], [28], [29], [34].

Most of the research regarding the spin use a similar approach in terms of the mathematical model and varies mostly on how to properly quantify the spin of the ball itself. By analysing one such research in detail, it is possible to understand the importance of quantifying the spin of the ball.

According to the results in the papers [3], [4] and [8], the equation that represents the flight of the table tennis ball is represented as follows:

$$\dot{V} = \frac{F}{m} = -\frac{1}{2m} C_D p A \|V\|V + \frac{1}{2m} C_M p A r \omega \times V + g \quad (2.1)$$

Where F is the total force on the ball, m is the mass of the ball, C_D is the coefficient of air resistance, p is the air density, A is the cross-sectional area of the ball, V is the velocity vector $V = [V_x V_y V_z]^T$, C_M represents the coefficient of Magnus force, r is the radius of the ball, $\omega = [\omega_x \omega_y \omega_z]^T$ is the angular velocity vector, g is the gravity and $\|V\|$ denotes the length of vector V .

The above relationship between the spin velocity vector and the lateral velocity vector can be converted to a discretised motion model, which makes the assumption that the sample period is very short [57], the discretised relationship with a sampling period T_s is expressed as follows:

$$\begin{bmatrix} V_x(k+1) + k_v v_x(k) \\ V_y(k+1) + k_v v_y(k) \\ V_z(k+1) + k_v v_z(k) + gT_s \end{bmatrix} = \begin{bmatrix} 0 & k_m v_z(k)T_s & -k_m v_y(k)T_s \\ -k_m v_z(k)T_s & 0 & k_m v_x(k)T_s \\ k_m v_y(k)T_s & -k_m v_x(k)T_s & 0 \end{bmatrix} \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} \quad (2.2)$$

$$k_v = (k_d ||V(k)||T_s - 1) \quad (2.3)$$

And due to the spin velocity being perpendicular to the Magnus force:

$$\begin{bmatrix} V_x(k+1) + k_v v_x(k) \\ V_y(k+1) + k_v v_y(k) \\ V_z(k+1) + k_v v_z(k) + gT_s \end{bmatrix} \times \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = 0 \quad (2.4)$$

It is clear that the spin of the ball could potentially play a major role in defining the trajectory of the ball. The main issue is how can the spin be detected and quantified.

2.3 Ball detection and prediction

The most common method to track the ball is through optical computer vision. This method involves a setup of cameras that record the ball while it is moving in the air, process the images obtained from the cameras and find the ball in the current frame. The exact method to track the ball varies, as in [13], [17], [24], [35], but due to the nature of the ball, detecting the spin stays to be a challenge since the texture of the ball is homogenous and that makes it harder to sense whether the ball is spinning.

The main issue of ball tracking by cameras is that the ball is moving at a considerable speed and hence the frame rate is increased while the image resolution is lowered. If the ball spin needs to be carefully measured then the resolution should take priority while the frame rate is to be lowered [21].

Due to these issues, several research has been done specifically to find an efficient way to detect and quantify the spin of the ball [24], [27], [34], some of the methods propose to add a marker to the ball to make it easier to notice the spin [1]. While this approach proves to work, in a real-life situation it might not be entirely possible to implement due to game rules and the specific details needed for it to work properly.

However, the above reference in [1] is of some significance due to reaching an important conclusion that would lead in part to the basis of the solution presented in this thesis.

$$\begin{pmatrix} \dot{V}_x \\ \dot{V}_y \\ \dot{V}_z \end{pmatrix} = \begin{bmatrix} -k_d ||V|| V_x + k_m (\omega_y V_z - \omega_z V_y) \\ -k_d ||V|| V_y + k_m (\omega_z V_x - \omega_x V_z) \\ -k_d ||V|| V_z + k_m (\omega_x V_y - \omega_y V_x) - g \end{bmatrix} \quad (2.5)$$

The kinematic model of the ball in mid-flight (2.5) can be obtained from the previous equations (2.4) and (2.3), as explained in the paper [1]. Since this model is only describing the interaction between the ball and the air, it needs to be complemented with another set of equations describing the behaviour when the ball hits the table, the derive process is fully explained in paper [2].

$$\begin{cases} V_{xout} = [V_{xin} \omega_{yin}] b_1 \\ V_{yout} = [V_{yin} \omega_{xin}] b_2 \\ V_{zout} = V_{zin} b_3 \\ \omega_{xout} = [V_{yin} \omega_{xin}] b_4 \\ \omega_{yout} = [V_{xin} \omega_{yin}] b_5 \\ \omega_{zout} = \omega_{zin} b_6 \end{cases} \quad (2.6)$$

Where $b_1, b_2, b_4, b_5 \in \mathbb{R}^2$ and $b_3, b_6 \in \mathbb{R}$. This indicates that the spin of the ball, ω , is directly correlated to the speed in all its axis when flying and bouncing on the table, if we analyse the effect of the spin after it hits the table, it can be realised that the equation to calculate the spin, ω_{out} , is very similar to the equation to obtain the velocity [30]. Since there are no extra factors to take into account (such as an external force like the air resistance affecting both the velocity and the spin when they hit the table), it can be proven that the first few moments after the hit are crucial in order to predict the trajectory and the bouncing of the ball on the table [72], [73]. To get a decent prediction it is sufficient to take camera measurements shortly after the ball hits the table as per some research have demonstrated previously [23], [18].

2.4 Control of the ball

After being able to predict the trajectory of the ball, the next step is to interact with it through the bat. This interaction will be the one controlling the ball and giving it a new trajectory that we should be able to define through the positioning of the bat.

Most current table tennis robots follow the same approach in order to accomplish this control over the ball [50]. The analysis of past research, for example [9], explaining a methodology to design and control the landing points for a competitive table tennis robot, it is possible to understand the basic methodology necessary to control the ball. There is no need to go into the details of the intricacies of the ball control as the focus here is in the broader methodology. Figure [2.4.1] outlines the principles of the methodology in [9] as built on well-established techniques.

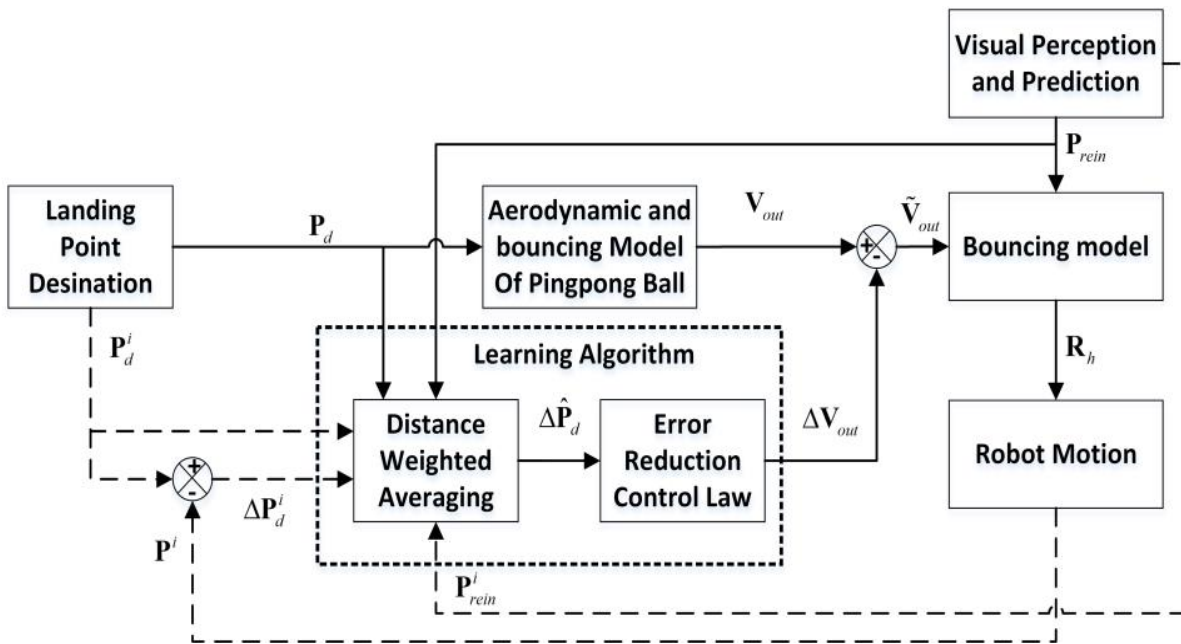



Figure 2.4.1 Example of a typical control method for a table tennis robot [9].

It can be noted that the novelty of the research presented relies on the learning algorithm being implemented as a complement to the current system rather than replacing any step of the method. The approach taken to control the ball is proposed to be the same as in [9]. In essence one should first sense the ball trajectory in the past, until shortly after the bounce, and predict its future trajectory




before the controlled hit. This is in essence “visual perception and prediction“ to create a model to predict the trajectory once it hits the bat, followed by known methods of bat control.

Every researcher might have a different methodology to achieve the control of the ball but, in a general situation, the approach previously explained is the one used in most research [4], [8], [14], [16], [18], [24], [25], [31], [43], [64], to control the ball.

2.5 Machine learning in table tennis

Machine learning is a set of algorithms in control sciences that has been used in the last few years in many applications with varying degrees of success. These algorithms are capable of performing different types of tasks and most of them are flexible in their approach when solving a problem.

Machine learning has been applied to some extent to table tennis [32], [37], [53] and the results so far have been promising. However, if we analyse the results obtained in some of these papers [86], [87], while they were able to accomplish their objective, the problem was proposed as linear in nature and thus the results can only apply if the problem remains limited to that linearity. It has been established through equation (2.6) that the spin of the ball has a direct correlation to the velocity of the ball so when the spin is present, which is an average occurrence in non-basic nor entry level table tennis game, the problem needs to be defined as nonlinear. This nonlinearity of the problem is still pending to be properly addressed in a machine learning approach and is causing a potential problem yet to be properly examined.



2.5.1 Machine learning applied to predicting the ball trajectory

One of the main areas where a machine learning approach is being applied is in the prediction of the ball. This is due to the prediction of the ball being the first stage to control it.


An example of this research can be appreciated in the paper that managed to control the ball through a regression neural network [86].

The results in this research [86] manage to successfully confirm that machine learning is a viable tool to generate a prediction of the trajectory of the ball. However, it can be appreciated that the tool is only used to predict the final ball position, which is the main objective in a prediction problem, but for classic control purposes it would need more information in order to build up a system capable of handling the ball, such as speed and acceleration. Thus, this approach, while useful, confirms that it would need a new control system compatible with machine learning in order to work as intended.

The main issue with this research is that a neural network is being used to propose the mathematical equation for a classic predictive control approach, which is still bounded by certain limitations such as having to compute the equation as a sum of serialised finite number of equations, which for a machine learning based approach it might not be a necessary step.

2.5.2 Machine learning applied to controlling the ball trajectory

In recent years, research has accomplished the first steps towards controlling the ball trajectory using a machine learning approach. One of the most important research projects that managed to do so, used a reinforcement learning methodology [88].



Most of these new approaches [61], [66], [68] are focusing exclusively on the control of the ball, which yields good results but in some of the tests, the robot was not able to properly hit the ball back as expected due to it not positioning itself correctly, which happened due to an incorrect prediction of the ball.

It is important to emphasise that the control of the ball should be a priority only if the prediction of the ball position is already accurate enough and therefore this research will focus entirely on the prediction of the ball.



Chapter 3

Technical preliminaries

In this chapter, the theories, and tools necessary to carry out the experiments will be discussed and explained, and they will emphasise how they will be applied in the next chapter.

3.1 Machine learning

Within control sciences, and within that in artificial intelligence, machine learning is a set of algorithms to improve performance of feedback control systems. This is achieved through algorithms capable of adapting themselves and improving their own performance based on data they collect.

Machine learning algorithms are also capable of finding patterns in a database, which then are processed to obtain a decision and a prediction of the solution, as part of the feedback control processes. The method these algorithms operate varies from algorithm to algorithm but they can be roughly divided into two groups of techniques.



3.1.1 Unsupervised learning

This technique is applied when the data needs to be explored, the objective is not particularly set and the pertinent information in order to achieve the objective has not been properly extracted. This method is mostly used in order to reduce the amount of data and find probable patterns.


Since this problem have a defined objective and the data is already fit for purpose, as evidenced in previous research [19], [36], [58], [89], this technique is not suitable for this research.

3.1.2 Supervised learning

These algorithms require an established set of input and output data, which then it uses to train a model to generate predictions to be able accommodate a new set of data.

The main difference with unsupervised learning is that in this case the algorithms will perform as a method of classification, a tool capable of predicting discrete responses, and regression, prediction of continuous responses.

Within the classification algorithms some of the most common tools are logistic regression, k-nearest neighbour, super vector machine, neural network, naïve Bayes, genetic programming, discriminant analysis, decision tree and bagged and boosted decision trees, the details for each tool can be find in specialised books and research on neural networks [74], [78].



The most common examples of regression algorithms are linear regression, nonlinear regression, Gaussian process regression model, support vector machine regression, generalised linear model and regression tree.


For this research, only two of the most common and suitable algorithms will be selected to make an analysis of both methods and discover the weaknesses and potential strengths of the approaches once applied to the problem.

3.1.3 Genetic programming

The heuristic method chosen to replace the mathematical model in the prediction of the ball trajectory was genetic programming. This tool was chosen over other algorithms because in other research [44], [83] with similar elements, its effectiveness has already been proven to be able to optimize functions, however it had not been used to propose an approach of this magnitude.

The idea on which genetic programming operates is basically the genetic evolution of living organisms, which dictates that a living being will evolve to better suit its environment and any organism that fails to do so is destined to perish. This simple natural rule is then transferred to a computational field and in the case of this research it can be understood that the proposed replacement of the mathematical model must be "evolved" until it can be considered suitable, which means that it is applicable to our system and that it performs its function with an acceptable degree of precision.

Since genetic programming performs similarly to how genes in real life act, the methodology of its function can be explained as the following:


-
- 
1. Generate an initial population.
 2. While the termination criteria is not met, which I will detail later:
 - a) Select individuals, which in this case are functions, for reproduction and elimination, considering their quality.
 - b) Combine and / or vary new individuals.
 - c) Add and remove individuals to generate a new population.

In this particular case, the functions, or individuals, are represented as trees which are made up of 2 elements:

- Terminals: Elements composed of the possible inputs to the individual, that is, the number of variables that the function will consist of.
- Parity functions: It is composed of the operators and functions that can compose an individual. For example: Boolean functions: and, or, not, xor. Arithmetic functions: addition, subtraction, multiplication. Conditional statements: if, then, else, case, switch, etc.

For this research, I used arithmetic functions and I tried to make them simple functions so as not to complicate the equation that will replace the mathematical model and that it was computationally easy to calculate, besides that it is not convenient to use a large number of functions, because this increases the size of the search space and could encourage more complicated processing. Once the terminals and the parity functions have been assigned, the following steps should be met in order to apply genetic programming:

Initialization: It consists of forming the initial population of individuals. One of the main parameters is the maximum size of the resulting tree. This limit can be




imposed on the number of nodes or on the depth of the tree, usually, two methods are used to generate this population, the growth method and the complete one.

- The growth method: Let T be the set of terminals and F the set of functions. An element of F is randomly chosen to form the root of the tree. The content of the child nodes of the root is chosen from $F \cup T$. If the chosen value is a function, this procedure is repeated with the children but if the chosen value is a constant, that branch of the tree is terminated.
- The complete method: The full method grows the tree similarly to the grow method, but elements are always chosen from the set of functions, unless the node is at maximum depth, in which case only elements of T are chosen. The method results in balanced trees of maximum depth.

If the number of nodes is used as a size limit, growth is terminated when the tree size has reached the limit. Once the populations are initialized and the method to grow the population is defined, the next step is to design the genetic operators, which are the ones that will begin to vary the individuals. The most common operators are:

- **Reproduction:** It is the simplest operator of all. An individual is selected and duplicated, leaving two copies within the population.
- **Crossover:** The crossover operator combines the genetic material of two individuals by exchanging pieces among the parents in order to produce two descendants, a node is then randomly chosen from each tree and then the subtrees under these nodes are exchanged.


-
- 
- **Mutation:** It is applied on a single individual and acts with a probability, in general, very low and that is a parameter that can be calibrated. There are many types of mutation, for example:
 - Point mutation: A single node is exchanged for another of the same class.
 - Permutation: The arguments of a node are permuted.
 - Survey: New individual is generated from a subtree.
 - Expansion: A terminal is exchanged for a randomly generated tree.
 - Collapse: Subtree is exchanged for a terminal.
 - Subtree Mutation: Subtree is replaced by another randomly generated one.
 - Gene Duplication: Subtree is replaced by a random terminal.

Quality function: Quality is the measure used by genetic programming, during simulated evolution, to measure how good a solution is. In this case, checking if the final equation meets the requirements to be suitable replacement for the mathematical model. In order to achieve this objective, a selection process needs to take place.

Selection: It is the process by which individuals are adopted from one generation to another, there are different types of selection, the first of them is based on genetic algorithms:

For a population of "N" individuals:

1. Choose two parent individuals, favouring those with the best quality.
2. Apply crossover
3. Apply mutation
4. Reproduce
5. Repeat until a new generation of "N" individuals is completed



Another technique is to run tournaments:


1. Choose two groups of n individuals randomly from the population.
2. Select the best element from the first group, and the best element from the second.
3. Apply crossover
4. Apply mutation.
5. The two new individuals replace the worst of each of the groups.

This latter technique is generally preferred for reasons of efficiency. The quality function may be inspired by the current mathematical model and use a very similar equation, this way it is ensured that the result is similar to this function but with some variations. It is only needed to assign one function as a quality function in order to check that the resulting new equation from the subtree process is effectively convergent and the same will be applied to each function in order to compare the results and check which one gives a better convergence.

The stop condition is when the best individual in the population has an acceptable quality, where quality is determined by how close the equation is to converging in comparison to the original mathematical model equation.

3.1.4 Neural networks

A neural network is an interconnected set of simple processing elements, units or nodes, whose functionality has some similarity on the animal neuron. The processing ability of the network is stored in the connections between units, called weights. These are obtained through an adaptation or learning process from a training set.



Neural networks are a massively parallel distributed processor constructed of simple processing units that is naturally prone to storing experimental knowledge and making it available for use.


An artificial equivalent of the biological neuron is the threshold logical unit, while the synapses are modelled as scalar weights. Each weight multiplies by one input before being sent to the equivalent of the cell body. In consequence, all the products are added arithmetically to give as a result the activation of the node.

The activation is compared to a threshold, if the activation exceeds this threshold, the threshold logical unit produces a 1, otherwise it returns a 0. There are other types of activation methods but the one described earlier is among the most linear. Each entry is multiplied by a weight before being sent to the equivalent of the cell body through the neuron.

The term "network" will be used to designate any system of artificial neurons, from a simple node to an entire arrangement of nodes interconnected in some way between them. One typical architecture is the layered arrangement, where nodes in one layer connect forward with neurons in the next layer. This type of arrangement is just one of diverse approaches in which the nodes in a network can be interconnected.

In the case of real neurons, their synaptic forces can be modified according to the input stimuli as needed, in the case of an artificial neuron this is manifested in the modification of the values of the weights.

In terms of information processing, the system starts from a default state stored in weights, which can change through an adaptation process based on stimulations obtained from a pattern set.



One mechanism of adaptation of the weights is through what is known as supervised learning, which means that an input pattern is presented at the input of the neural array and the response of the network is compared to a specific target output. The difference between the network output and the target output determines how the network weights are modified.

Each method intended to modify the weights constitutes a learning rule, which work under the approach that after adjusting the weights, another new input pattern is presented to the network and the new changes to the weights are made. This sequence is repeated in an iterative manner until the behaviour of the network converges. This whole process of adjusting weights is known as a learning algorithm.

If the network has learned the structure of the problem in question, the network should correctly classify the new input pattern. This characteristic is known as the *generalizability of the network*.

3.1.5 Threshold Logic Unit

The threshold logic unit may have a finite number of inputs, x_1, x_2, \dots, x_n , and the effect of each synapse is modelled by a number or weights, $\omega_1, \omega_2, \dots, \omega_n$, these weights will amplify the input of each neuron, x_n , separately. Resulting in the following:

$$TLU = [\omega_1 x_1, \omega_2 x_2, \dots, \omega_n x_n] \quad (3.1)$$

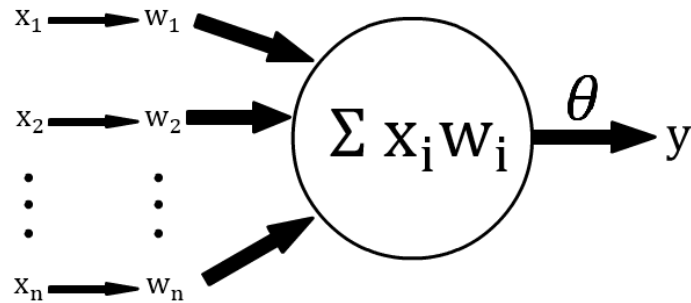


Figure 3.1.5.1 Graphical example of a typical Threshold Logic Unit.

The excitation and inhibition actions are modelled, respectively by positive or negative values. Each of these products can be positive or negative depending on the sign of the weight.

These products are combined trying to emulate the process that occurs in the axon, in the case of threshold logic unit, a , this is done simply by adding the products, resulting in the activation of the corresponding neuron:

$$a = \omega_1 x_1 + \omega_2 x_2 + \dots + \omega_n x_n \quad (3.2)$$

The result of the neuron activation, y , can then be understood to have two main types of responses, one of them is the step response, in which a threshold, θ , needs to be defined and when this threshold is reached through the summation of the inputs, the response will be immediate.

$$a = \sum_{i=1}^n \omega_i x_i \quad y = \begin{cases} 1 & \text{if } a \geq \theta \\ 0 & \text{if } a < \theta \end{cases} \quad (3.3)$$

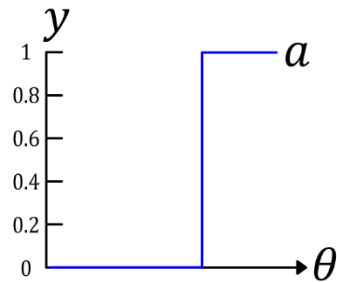


Figure 3.1.5.2 Graphical example of a step-up response.

The second type of response is the sigmoid type. This type of response is capable of obtaining a smoother signal by using the following equation:

$$y = \sigma(a) \equiv \frac{1}{1 + e^{-\frac{a-\theta}{p}}} \quad (3.4)$$

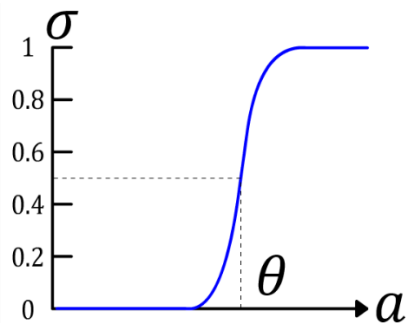


Figure 3.1.5.3 Graphical example of a sigmoid response.

Where p determine the form of the sigmoid, meaning that higher values will flatten the sigmoid response, while smaller values will make it resemble a step response.

3.1.6 Perceptron

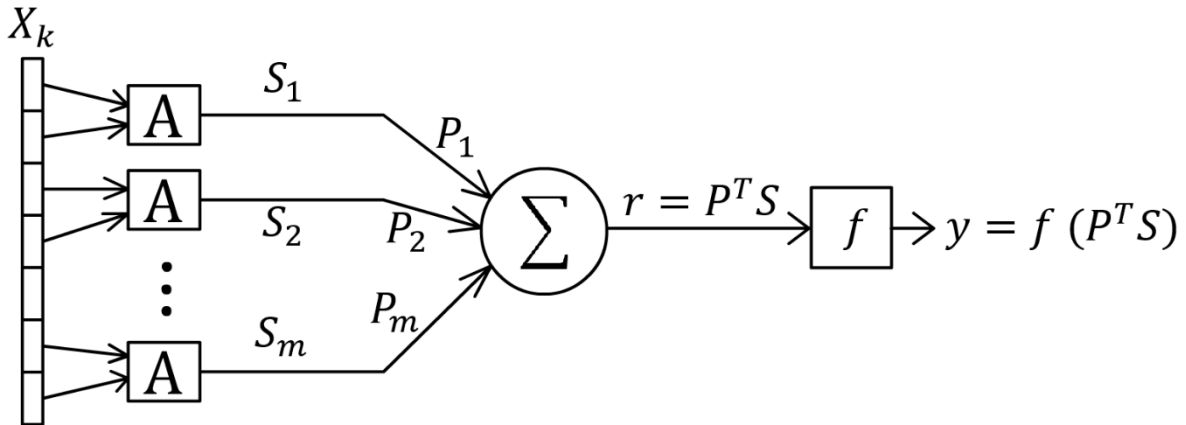


Figure 3.1.6.1 Graphical example of a perceptron.

The perceptron is an enhancement of the TLU, it works very similar, but the main difference is that it consists of sensing units, S , association units, A , and a processing and response unit, r . An Stype unit emits a 1 if energized. A randomly selected set of S units are connected to type A units. Each unit A accepts a certain number of m inputs and calculates a weighted sum, the weights can take the values $+1$ or -1 , and are assigned randomly. The sum is compared with a threshold θ resulting in an output of 0 or 1.

$$S_j = \begin{cases} 1 & \text{if } \sum_{k=1}^n P_k x_k \geq u \\ 0 & \text{if } \sum_{k=1}^n P_k x_k < u \end{cases} \quad (3.5)$$

The threshold θ for all A units is always the same one, the output of the k -th unit A is multiplied by a weight p , and a sum of all m weighted outputs is formed into a unit sum, where each weight p can be -1 , 0 , or 1 . The output of this sum unit is 1 or 0, depending on the threshold θ . The design of the perceptron involves adjusting

the weights P_k , as well as the threshold θ . It can be appreciated that the TLU is basically a special case of the perceptron with A units with only one input. In order to adjust the weight, it is advisable to use the delta rule. The delta rule can be explained as following.

With x_k as the input, it is possible to obtain:

$$\frac{\delta e^p}{\delta w_i} \quad (3.6)$$

Where e^p represents the error found in gradient descent and is expressed as:

$$e^p = \frac{(t^p - a^p)^2}{2} \quad (3.7)$$

Where t^p is a desired class, or result from our neuron. With this, it's possible to estimate the error, δE , on the weights, δw_i , by proving:

$$\frac{\delta e^p}{\delta w_i} = -(t^p - a^p)x_i^p \quad (3.8)$$

By substituting:

$$\Delta w_i = -\alpha \frac{\delta E}{\delta w_i}, \quad \Delta w_i = \alpha (t^p - a^p)x_i^p \quad (3.9)$$

For smaller values in, α , the result will converge, this means the weight vector will be closer to the one we require and the error will be minimum.

3.1.7 Networks with multiple layers

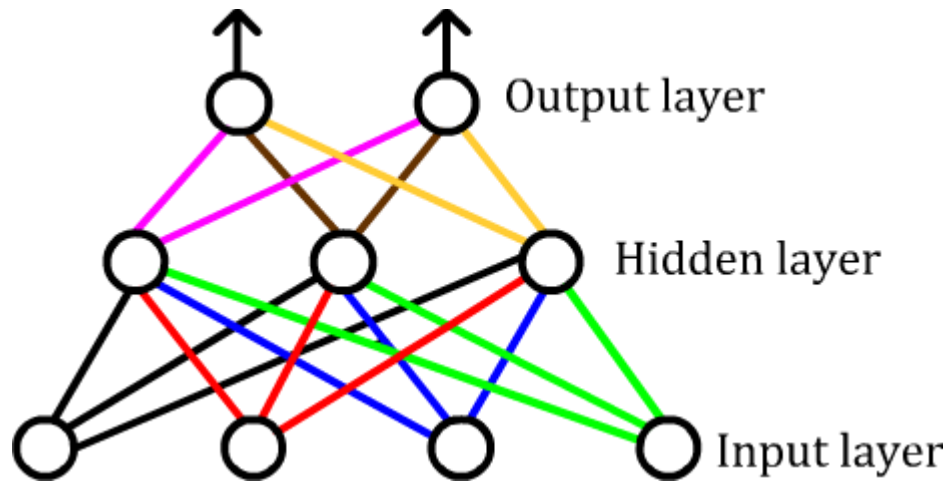


Figure 3.1.7.1 Graphical example of a multilayer network.

The input layer is a distribution point for the input signals, the hidden and output layers are made up of semi-linear nodes. The nodes of the hidden layer are so called, due to the fact that there is no direct access to the information of their outputs for the purposes of their training, its main function is to collectively form its own representation of the set of input vectors. The idea of the training is to use the descent of the error gradient as a function of the weights.

The error for networks with semi-linear nodes is given as:

$$e^p = \frac{1}{2} \sum_{j=1}^M (t^p - a^p)^2 \quad (3.10)$$

The calculation of the increments in the weights of the nodes of the output layer is similar to the case of a single-layer network, using the delta rule as per equation (3.9):

$$\Delta w_i = \alpha \sigma(a_j)(t^p - a^p)x_i^p \quad (3.11)$$

The index j refers to any of the output nodes in the output layer.

Since the nodes of the hidden layer are also semi-linear, then in the equation for the increase of the weights, the following terms must be incorporated:

$$\alpha \sigma(a_j)x_i^p \quad (3.12)$$

The term that should change is:

$$(t^p - a^p) \quad (3.13)$$

This is due to the expression referring to the comparison of the output of the output layer nodes and the membership values t^p of the input vectors x_p . The following equation is then proposed:

$$\Delta w_{ki} = \alpha \sigma(a_k)\delta^p x_i^p \quad (3.14)$$

The expression δ^p is the one that needs to be defined. Since we are now working with a hidden layer, the nodes will also need to be expressed differently in order to understand how they affect each other, it is proposed to assign k as the nodes from the hidden layer and j as the nodes from the output layer, which means we now have w_{kj} representing the weight of the hidden layer.

The effect that node k has on the error will depend on two factors:

- 1) How much node k influences the output of node j .
- 2) How much node j affects the error.

The more k affects j , the greater the effect on the error and, j error will also be in its own difference expressed as δ^j . It is important to mention that the effect of k on j depends on w_{kj} and the interaction between these two factors is given by their product $\delta^j w_{kj}$

Since k is affecting several other nodes on the output layer, then δ^k can be represented as the sum of all its products.

$$\delta^k = \sum_{j \in I_k} \delta^j w_{jk} \quad (3.15)$$

Where I_k is the set of nodes in the output layer that receive output from node k . These are then the equations needed to calculate the weights for the hidden layer nodes:

$$\Delta w_{ki} = \alpha \delta^k x_i^p \quad (3.16)$$

For the hidden layer nodes:

$$\delta^k = \sigma(a_k) \sum_{j \in I_k} \delta^j w_{jk} \quad (3.17)$$

For the output layer nodes:

$$\delta^k = \sigma(a_j)(t_k^p - y_k^p) \quad (3.18)$$

In order to obtain the error in these networks, it is required to use a back propagation algorithm. The algorithm is very similar to the delta rule and can be explained as following:

Back propagation	
Step forward	Present the pattern to the input layer.
	Evaluate the outputs of the hidden layer nodes using the input patterns.
	Evaluate the outputs of the output layer nodes using the results obtained in the previous step.
Step back	Apply the desired patterns to the output layer nodes.
	Calculate δ^k for the output layer nodes.
	Train each output layer node.
	Calculate δ^k for the hidden layer nodes.
	Train each hidden layer node with Δw_{ki}

Table 3.1.7.1 Table representing the steps to apply the back propagation method on a hidden layer network.

It can be observed that the first three steps require to make the calculation of the weights in a feed forward method, while in the rest of the steps it is necessary to do it backwards. In the last two steps, it can be appreciated that δ^k is propagating

from the output layer nodes to all of the hidden layer nodes, this is the reason why this algorithm is called back propagation.

3.1.8 Convolutional neural network

The convolutional neural networks are a multilayer artificial neural networks or deep neural network. The architecture of a convolutional neural network is composed of multiple types of layers that fulfil different types of functions; these types of layers are:


Convolutional layer

The convolutional layer is the layer which receives the input image and perform the convolution operation. This operation can be explained as a convolution filter, also known as kernel, which is applied to the input image and consists of the following:

$$C_{xy} = \sum_{i=1}^n \sum_{j=1}^n w_{ij} I(x+i-1, y+j-1), \forall x, y \in \{m\} \quad (3.19)$$

Where n is the value of the kernel to be applied, m is the value of the feature map, w_{ij} is the component of the convolution filter and $I(x, y)$ is the intensity value of the pixel (x, y) in the image I .

In general, in each convolution layer it is possible to apply any number of filters and obtaining a feature map for each one. In addition, it is possible to use multiple input images, which must be processed by three-dimensional convolution filters, where the depth of these filters must match the number of images in the




input. The activation functions commonly used in these layers consist of linear functions, such as the rectifier linear uniform function defined as:

$$f(x) = \max(0, x) \quad (3.20)$$

Since the architecture of convolutional networks is usually composed of multiple convolution layers, during the back propagation algorithm, the derivative of the function with respect to the weights of the layers closest to the input of the network depends on the product of the derivatives of the following layers. This is because the derivatives are calculated by the chain rule, so the output layer depends solely on its derivative, the hidden layer that precedes it depends on its derivative and the output layer. By using functions such as the sigmoidal and hyperbolic tangent, the derivative is bounded in the range of 0 and 1, which results in the derivatives getting flattened on the most superficial layers of the network. Using a linear function allows to mitigate this problem in deep neural networks, since its derivative is constant by intervals.

Pooling layer

The pooling layer, also known as the sub sampling layer, is an optional layer that is applied to the feature maps obtained from the convolutional layer. This layer applies, through a sliding window, a function that subtract the most important information of the feature maps. Examples of commonly used functions in this layer are max pooling, which consists of taking the largest element of the window, and the average of the window elements. Applying this layer brings multiple advantages, among which are:

-
- 
- Reduction of the dimensionality of the feature maps, which results in a reduction of the parameters to be trained in the network, in addition to helping to control over fitting.
 - Increases the invariance over small alterations, such as translation and rotation, when representing multiple local patches under an average or maximum value.


Dropout layer

The dropout layer is an optional layer that can be applied after most layers in the network. Given a parameter m , the dropout layer will be deactivating, with probability m of success, each of the neurons of the previous layer. This results in that, on average, only a fraction m of the neurons are used during each training stage. Although this is sometimes referred to as an additional layer in the literature [70], [75], [76], [84], [79], strictly speaking it is not, since the regularization of parameters occurs at the level of the connections between layers.

The main objective of this layer is to reduce the over fitting of the model by using fewer parameters than the total of the network during the training stage, which also results in shorter training times.

Dense layer

Since the output delivered by the convolution and pooling layers consists of a features map, which can be interpreted as the application of one or multiple filters to the input image, these maps are often used as input for other neural networks. A



dense layer consists of a feed forward network, whose vector of inputs corresponds to the vectorization of the features map obtained in the previous layers.


These networks are trained over multiple epochs. During each epoch, the network receives the input data and applies the training algorithm to adjust its weights. At the end of each epoch, the performance of the network is evaluated and the learning process stops if the convergence condition is reached. In general, the training stop condition consist of observing negligible changes in the network prediction error for the training data or setting a maximum number of iterations.

3.2 Machine learning approach comparison

Each machine learning tool has different advantages, disadvantages, and requirements to solve a particular problem. While some problems allow to try a different approach, some others are more complicated or limited.

In terms of table tennis, the approach to solve a particular problem depends entirely on the researcher itself, there are plenty of examples in regard to predicting the ball trajectory alone [3], [4], [16], [21], but for this particular research, two tools are selected due to previous successful attempts at solving similar problems like the one presented in this thesis.

The first one is genetic programming. The main reason to use genetic programming is due to its ability to propose equations and diverse numerical methods to fit an area under a curve or simply emulate the behaviour of an observer pattern. Perhaps the main disadvantage is that it can take a very long time to find the solution and it does not always manage to find a global or convergent solution, but instead just a local one. Another disadvantage is that in order to work efficiently,



a fitness function need to be proposed, which in some cases might defeat the purpose of allowing the machine learning technique to come up with a real self-adjusting methodology [71], [77], [81].

The second tool selected is neural networks. Neural networks have the main advantage of being simple to apply to a wide range of problems and the documentation regarding each of these attempts is, for the most part, extensive in comparison to other machine learning tools [46], [48], [52], [54], [55], [59]. Another advantage it presents, relies on being able to work directly with images that are already similar to the ones obtained through the cameras used in the current prediction approaches.

3.3 Object tracking

There are several approaches to track an object through computer vision [11], [20], [49], [88], most of these methods depend entirely on the specific objective required. While some tracking methods make emphasis on the accuracy of the recognition of the object [12], others might focus on the speed of the tracking [89].

For this research, the objective must focus on tracking the object as fast as possible due to the time constraints present in the problem itself. It is then proposed that a meta heuristic approach should be considered due to the nature of the specific parameters needed to work in this particular setting.

Another issue to take into account is the probability of needing to obtain a usable database for the machine learning approach directly from the tracking system itself, which can only be realised through a specific meta heuristic approach based on a similar system also focused on speed.



Chapter 4

Methodology

In this chapter, it is determined the approach in which the novel approach is applied, detailing why a particular method was preferred over different ones and performing several tests, in order to justify the reliability of the method to be employed.

4.1 Table tennis ball recording setup

The setup for this experiment is based on previous research tests with similar objectives regarding the prediction of the ball [3], [4], [6], [7], [8]. A setup with at least two or more cameras is used due to simplifying the problem of working in a 3D environment through segmentation of its hyper planes, which results in two or more linear systems that are easier to manage.

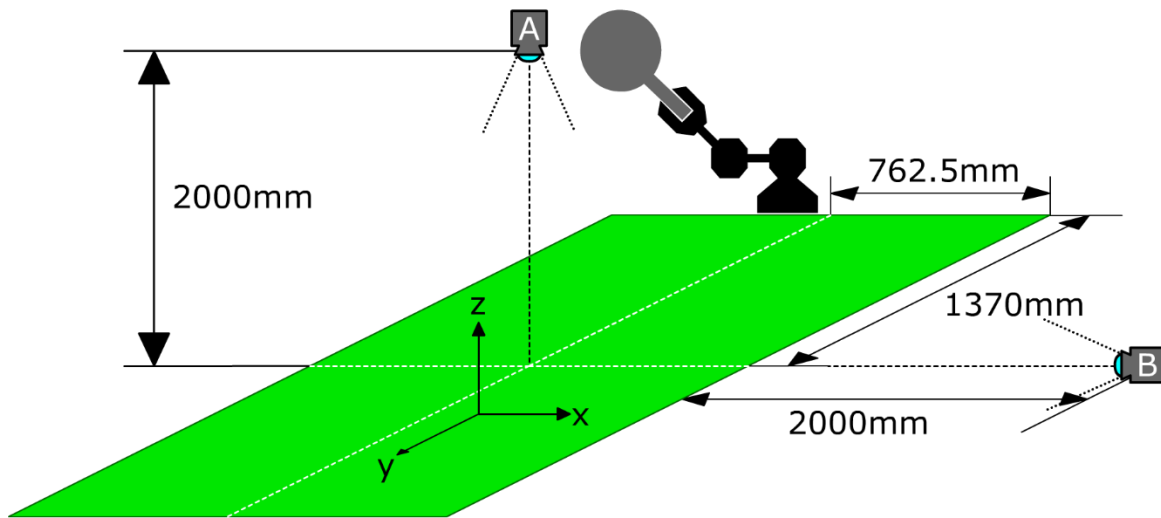



Figure 4.1.1 Proposed setup consisting of two cameras, A and B, distributed above and on one side of the table respectively.

To simplify the parameterization of the 3D coordinates, both cameras are located on the same y axis. The position of the lateral camera, B , may be switched to the opposite side if required, as long as the distances in relation to the y and z axis are kept unchanged.

The same model of camera was used in the setup to avoid further difficulties when processing the images. The reasoning behind different cameras in this approach is that each camera can work as an individual module running its own machine learning approach and it would only need to be properly synchronised when proposing a specific solution to the prediction of the ball in its hyper plane.

The tracking of the ball on the camera A is going to be different from the tracking approach on the camera B , this is due to the background elements being more challenging on one camera in comparison to the other. Camera A is going to be able to capture most of the table, which is solidly painted with an even layer of colour, resulting on a background which is easier to process. However, camera B is



going to capture more movement and background elements that might interfere with the tracking of the ball while on flight. To avoid discrepancy and high latency in just one of the cameras, a backdrop was implemented to reduce the number of resources required to track the ball properly.


4.1.1 Raspberry Pi

A Raspberry Pi 4 B with a camera was used to act as a module for recording. The decision of using this particular hardware is due to the capacity of the board to process a decent number of calculations.

The camera used is an Arducam OV5647. The camera has an input clock frequency between 6 and 27 MHz with an adjustable transfer rate ranging from QSXGA, resolution of 2592x1944 pixels at 15 frames per second, to QVGA, resolution of 320x240 pixels at 120 frames per second. Since the background of the recorded image is proposed to be as even as possible in order to make the processing more efficient, the QVGA transfer rate with the highest frames per second has been selected for the remaining of the research.

The Raspberry Pi 4 B has a Quad core Cortex-A72 ARM 64bit processor with a speed up to 1.5Ghz, the boards used in this research have an 8Gb LPDDR4-3200 SDRAM memory. The operating system is a customised version of Raspbian is, this modified version is available through the Matlab website for free and the only noticeable modification is the addition of libraries and configuration of ports in order to allow Matlab to write and read data from the Raspberry Pi Ethernet port.

The flexibility to program on the Raspberry Pi board alongside its current processing capability, allows it to be a viable method to implement simple machine learning algorithms and the onboard camera decrease the time and necessity to



transfer the video stream and instead it can be processed directly on the board. This is the main reason why this approach was deemed the most viable for this research.

The connectivity between the Raspberry Pi board and a computer is achieved through the Ethernet port due to its speed and low probability of failure to connect. This method of connectivity is used for the remainder of the experiments to reduce the probability of stuttering, delay or communication failure between the modules.

4.1.2 Matlab

Matlab is a widely recognised software intended for numeric computing, one of its main features is the ability to download, create and customise toolboxes for specific type of problems. In the last few months, Matlab released a toolbox capable of controlling a Raspberry Pi board through the Ethernet port. The capability of being able to communicate directly to a Raspberry Pi board from Matlab makes it easier to apply the machine learning tools that currently exist in the Matlab community.

As stated previously, the probability of having a low latency is a main priority in this research due to the time constraints that the system is going to be operating on. It is due to this fact that several options were explored when testing the capability of handling two Raspberry Pi boards in synchrony and independently. The software that would be used to program the machine learning algorithm was also explored and three main options proved to be desirable, each of them had different advantages and disadvantages which are explored in the table below.

	Using each board separately	Synchronising each board to work as a group
OpenCV + Python	<ul style="list-style-type: none"> * Fastest to run the required algorithms * Multiple connection errors when attempting to transfer result of the machine learning algorithm 	<ul style="list-style-type: none"> * Multiple connection errors when connecting each module
ROS	<ul style="list-style-type: none"> * Delay on the camera when acquiring real time images 	<ul style="list-style-type: none"> * Severe delay when obtaining the current camera frames * Multiple connection errors when requesting camera feed
Matlab	<ul style="list-style-type: none"> * Delay on transferring data from the machine learning algorithm * Due to customised OS, not enough resources available to run proper machine learning algorithms 	<ul style="list-style-type: none"> * Occasional delay on data transferring

Table 4.1.2.1 Table comparing the main disadvantages observed when implementing different approaches with a simple machine learning exercise on the Raspberry Pi.

While OpenCV and Python might have been the fastest to successfully run the machine learning example on the board, its inability to export the data properly and reliably to the computer that would control the robot arm is the main reason why this approach was not chosen. Instead, the second most reliable approach is using the third option which would be Matlab.

Another advantage of Matlab is the availability of customised tools for specific robots, which can be applied directly in the function or the programming section that is already running the machine learning algorithm. The control of the robot will then be linked directly to the output of the machine learning toolbox, allowing it to respond as fast as possible but retaining the flexibility to modify any parameter needed within the specified Matlab customised tool for the robot.



4.2 Database for machine learning

The database will be developed through a set of stringed data with very specific characteristics which the machine learning tool is able to exploit in order to discern a particular pattern and create an algorithm capable of describing the database.

For this research, the database is obtained from real life data, it will be comprised of recordings of a table tennis ball being used in different settings. The settings will range from two human players playing against each other, a table tennis robot throwing balls at a defined landing point and at a particular area within a perpendicular wall from the table.

The setup for the recordings should be as close as possible between settings in order to avoid discrepancy errors that might arise due to an inconsistent data gathering and processing. The only exception to this particular suggestion is if the database will only be used for control purposes rather than prediction, in which case the database can differ from the setup as needed.

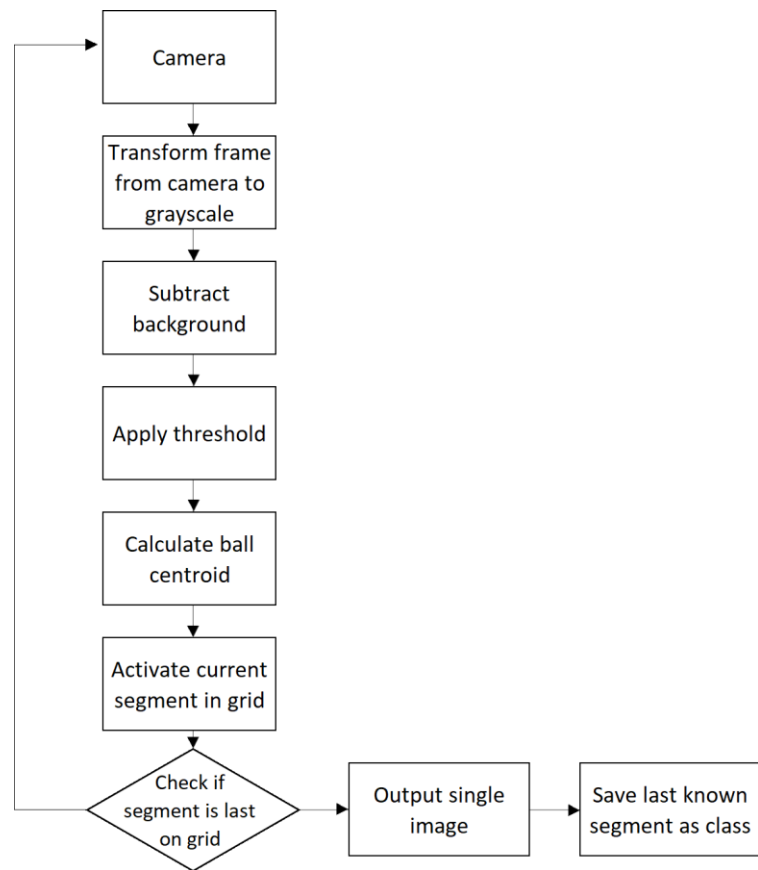


Figure 4.2.1 Proposed methodology to train the neural network.

This database is created through a novel method which involves a customised image processing and object tracking methodology which its main feature relies on obtaining an image as efficient and suitable as possible to make the machine learning process as optimised, in terms of speed, as possible.

4.2.1 Area of interest

Using the setup described in the Figure [4.1.1] and having the light source and the background defined, the area of interest for the machine learning tool is proposed. This area of interest can be defined as, *the minimum area within a picture representing the limits where the object of interest is capable of movement.*

This area of interest was defined manually before making any recording by taking a screenshot with the same resolution intended to use in the rest of the experiments and then crop the area accordingly.



Figure 4.2.1.1 Example on defining an area of interest on a screenshot.

The database required for the machine learning approach is a set of images with only the ball trajectory represented in each image. This means that each image should represent the ball trajectory from the moment it is being hit by the opponent, until it reaches a position where it is expected to meet the robot holding the bat. This final position may be defined depending on the type of robot used. For this particular research, the edge of the table is considered as the optimal point of contact between the ball and the bat.

4.2.2 Camera frames processing

A frame may be understood as an image taken directly from a camera, which can be represented as a function:

$$\begin{aligned} I: U &\rightarrow \{0,1,\dots,255\}^c \\ U &= [[0; m - 1]] \times [[0; n - 1]] \\ c &\in \{1,3\} \end{aligned} \quad (4.1)$$

Where U represents the pixels, m and n are the row and columns respectively, c is the number of colours. $I(i)$ is the i th pixel value in the image and where $i \in U$

$$c = 3, m = 240, n = 320$$

A subtraction of background elements is needed, to accomplish it, the current frames are transformed into greyscale, this is due to being preferable to work with a hyper plane representing a 2D matrix rather than a 3D set with vectors. To convert rgb to greyscale the NTSC formula was implemented:

$$I_g = \{(0.299 \times I_1) + (0.587 \times I_2) + (0.114 \times I_3)\}^c \quad (4.2)$$

Where I_g is the grayscale equivalent of I and I_1 is the red set, I_2 is the green set, I_3 is the blue set and $c = 1$

The background frame is stored as a function, I_{Fg} , which will contain all the information of the background noise that needs to be subtracted from the next set of

frames where the ball is going to be present. A tolerance, T , is used to amplify the subtraction of the background elements and get rid of the background noise.

$$I_{Fi} = I_{Fg} - (I_g + T) \quad (4.3)$$

$$T \in U$$

$$T = \{0,1,\dots,255\}$$

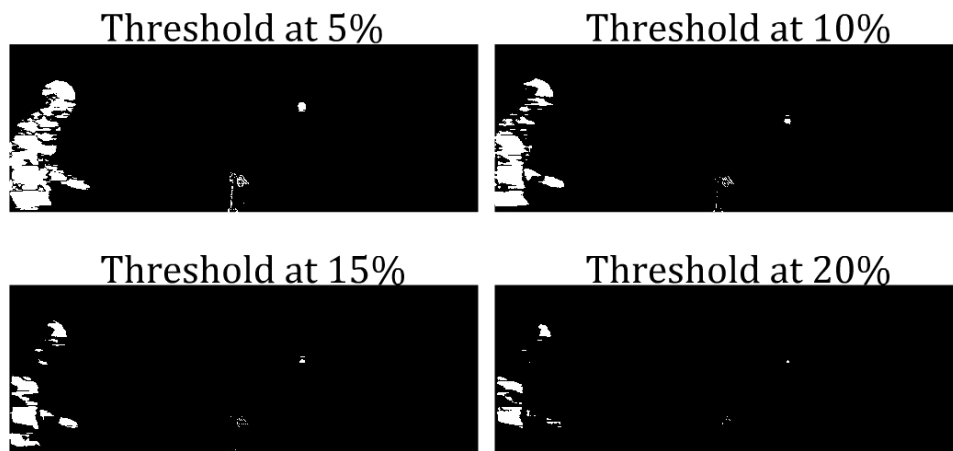


Figure 4.2.2.1 Graphical example of the results of different thresholds.

This tolerance enables the tracking algorithm to work under subtle changes of lighting which may occur when recording in settings that do not have a constant light source. The tolerance was defined through a metaheuristic approach inspired on maxima and minima. A tolerance of 15% of the background frame is chosen for the rest of the experiments.

Since the background is being subtracted, the majority of noise is minimal and the only objects present in the frame is the ball.

The ball itself has a round shape which can be represented in different ways when processing the current frame. In this research the ball is represented by a single particle which is calculated as the centroid of the ball.

$$c = \frac{1}{n} \sum_{i=1}^n x_i \quad (4.4)$$

Where n is the distinct points of x . For this research, it is proposed to use an approach similar to the single particle method, but with an important addition, instead of tracking the ball through the whole area of the frame, the frame will be divided into subsections and each subsection represent a smaller area of interest.

The purpose of segmenting the table is to standardize the distances between both cameras and simplify the process of 3D modelling. There are several methods to calculate the distance between the object and the cameras [38], [45], [56], [82], while these methods prove useful and give a precise and accurate distance approximation from the objects, in this research the accuracy and precision are lowered in favour of speed in order to move the robot arm as fast as possible. If the segmentation of the whole table, in all its axis, is realised in equal amounts on both cameras to form a grid on each hyper plane, the distance and the proportions of each segment will remain the same no matter the position of the camera. This particular method proves efficient when the camera setup tends to vary its position by small degrees of accuracy [40].

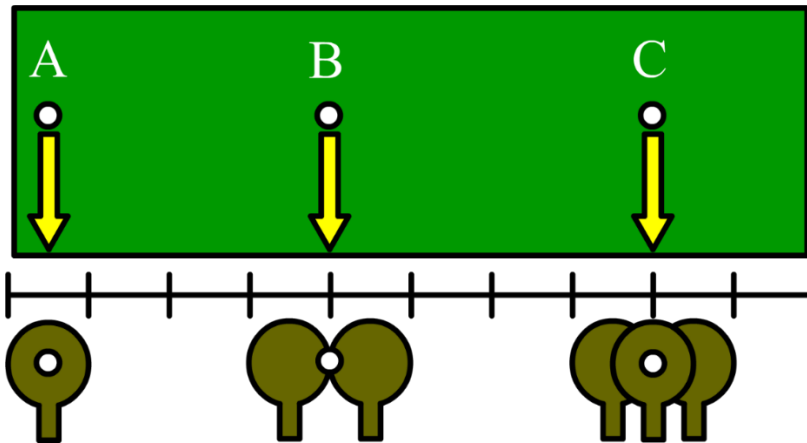



Figure 4.2.2.2 Table segmentation and ball contact area examples.

In the Figure [4.2.2.2] it can be appreciated a table divided in ten sections alongside three examples of an incoming ball and its contact area on the bat. In example *A*, the ball moves directly at the centre of one of the segments, in this case the bat positioned in the assigned section should have no problem having contact with the ball. Example *B* demonstrates the problem of an incoming ball between the limits of each segment, if the segments are not properly assigned, the contact area between the ball and the bat in any case of each segment is limited and can cause an error when trying to control its trajectory. Example *C* demonstrates that by adding an extra segment on the previous example is the most efficient approach to avoid uncertainties when the ball is incoming between segment limits. It is also important to mention that the number of segments should be carefully considered due to assigning too many segments might cause the system to slow down, the more segments available the more time it will take to track the ball trajectory in real life and the less time the machine learning method will have to properly classify the ball and assign the proper segment for the ball prediction.

For this research, the number of segments assigned to the table tennis are twenty segments per axis on all hyper planes.



Since the ball is currently represented as a single particle, due to the previous calculation of the centroid and now that the segments are defined, it is necessary to keep track of the segment where the ball is currently present. To do this, a flag is implemented so that every time the ball is spotted in one of the segments, this segment will retain the memory of the ball until the end of the tracking. As soon as the flag on a segment is activated, two actions take place.

1. The segment where the ball has been spotted should now be considered as the centre of a new area to crop the current frame, that is, using a similar approach to a k-neighbours approach, the inactive segments surrounding the current active one should now be the only ones considered in the next tracking iteration. This step is recommended to save time and computing resources.
2. If the segment being flagged is the last one in the grid, which means the ball is not going to be visible on the next iteration of the tracking, then the tracking algorithm end and the final image is sent as an output.

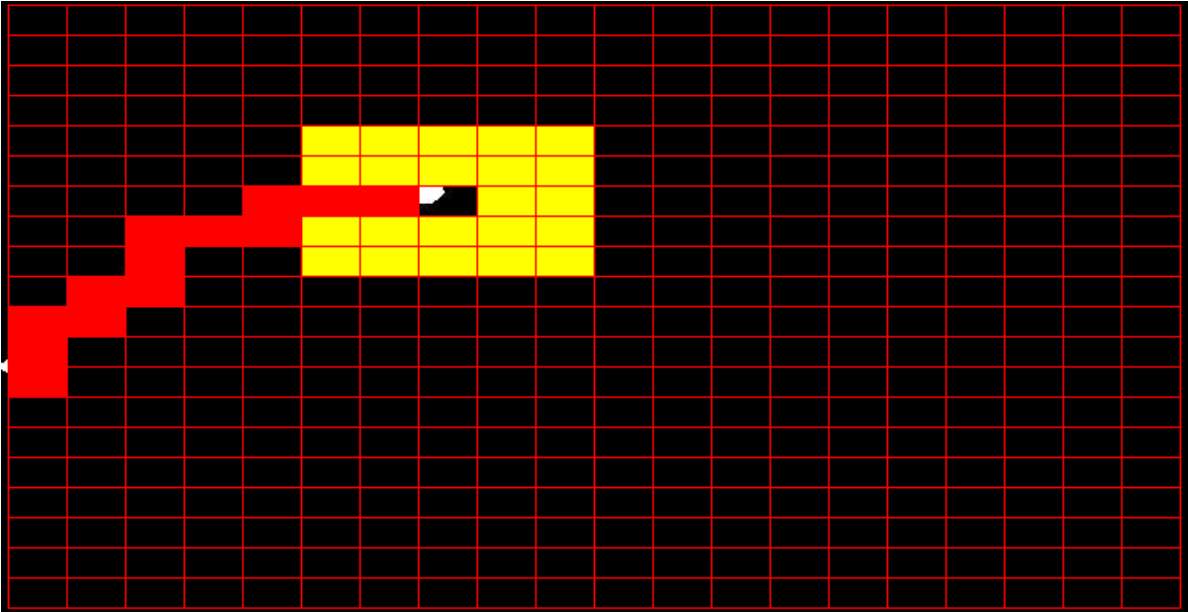


Figure 4.2.2.3 Graphical representation of the ball being detected on a frame with the grid in red. The yellow segments represent the segments activated due to the ball position in the grid. The red segments represent the segments where the ball has been detected previously.

4.2.3 Image output

When the tracking algorithm has finished due to the ball no longer being present in the grid, the resulting image now represent the trajectory that the ball had in regard to the segments of the grid. This trajectory is unique and is the one sent to the neural network either to train it or to classify it.

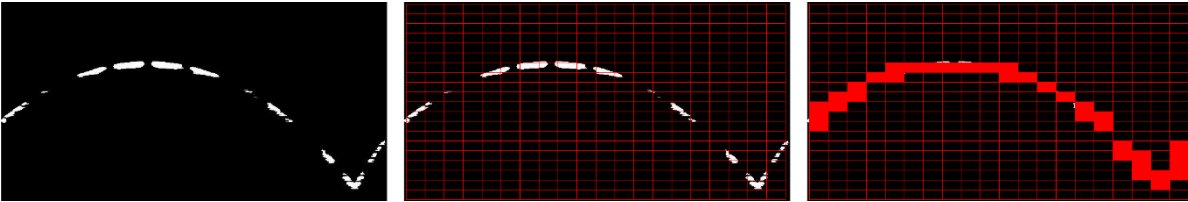


Figure 4.2.3.1 Graphical representation of the ball trajectory. Result of adding all the frames when the ball was being detected (left), applying the grid to the frame (middle) and flagging the segments where the ball was detected (right).



4.2.4 Convolutional neural network class


In case of training the neural network, the last segment flagged is considered the final position of the ball. When obtaining this trajectory, it is necessary to keep track of the full sequence of segments flagged and the last one because that information is the one that is going to be used to observe the patterns and define the outputs of the neural network.

The last segment flagged is always present in just one axis of the grid, this is the reason why the maximum number of classes, or labels, that the neural network will have, is twenty. It is not necessary to obtain examples of every single class when training the neural network, this is due to the nature of the game of table tennis itself, not every incoming ball will be possible to reply to either because it is going to land outside the area of the table or simply because the ball will end up in an unreachable location such as the lowest corner of the table.

4.3 Suitable machine learning tools

As detailed in the previous chapter, a supervised approach is chosen due to the problem having a defined objective and discernible pattern. From the supervised approach tools, genetic programming and a convolutional neural network were picked for the experiments due to different reasons.

Convolutional neural networks have been previously used in the literature for different sorts of image classification, which for this research proves to be a basic starting point due to the current research in table tennis using cameras systems to track the ball and perform most of the initial data gathering.



Genetic programming was chosen due to its ability to propose equations and fit areas under a curve, using an image as a base for the function that will be consequently applied in the algorithm. Due to the lack of literature in regard to this specific approach, this is a good reason to apply this tool as a novelty and to bring a point of comparison between other machine learning approaches.

4.3.1 Testing of the selected machine learning tools

Both tools were tested using a simulation of a table tennis ball with controlled variables. This control consisted in making sure the ball remained without spin, knowing the exact location, speed and acceleration at defined times and making sure the tools were able to solve the simple linear problem.

It is important to mention that the simulated table tennis ball will not take part any further in this research. This is due to the contradiction it would generate in the hypothesis which states that the current mathematical models in which the table tennis ball simulation is based on, is not entirely fit for purpose due to variables present in the real-life version of the game, like the spin of the ball, are not being properly implemented in the model.

4.4 Mathematical model replaced with genetic programming

For genetic programming, the chosen functions to be added as nodes are plus, minus, times, cos, and sin. The number of branches per node is maxed out to 2. The terminals are set to 40 and the operators will be crossover and mutation. The equation to be replaced in this instance will be just the aerodynamic model (2.1) such as the rest of the predictive methodology involving the bouncing model, robot motion and defining a landing point will remain intact.

4.4.1 Mathematical model replaced with convolutional neural network


For neural network, a convolutional neural network with a scaled conjugate gradient training method was chosen. The number of hidden layers and neurons will vary in order to compare the results and discuss the efficiency of the approach.

To replace the mathematical model with the novel approach, an evaluation of the weakness and fortitudes of the current methodology used to predict the ball trajectory and of a neural network is necessary.

	Weakness	Fortitude
Current prediction methodology	<ul style="list-style-type: none">• Not flexible• Functionality depends entirely on researcher experience	<ul style="list-style-type: none">• Highly accurate• Easy to modify values separately and evaluate results
Neural network	<ul style="list-style-type: none">• Requires very specific database• Can take long to train• Not as accurate as MPC	<ul style="list-style-type: none">• Flexible• Fast to compute

Table 4.4.1.1 Comparison of weakness and fortitudes of the current methodology to predict the ball final position against the proposed neural network approach.

A neural network excels at being a flexible approach with fast results, but to be able to compute as fast as it does, it needs to sacrifice accuracy. Current prediction methodologies, which are based on model predictive control, are more orthodox in their approach and while highly accurate [42], if the model proposed can perform efficiently, the dependency on the model might be the main issue.



To solve this potential problem, a novel approach was used, which consider the basics of model predictive control but exploit the main features of a neural network.

The current model to predict the ball trajectory is dependent on the serialisation of one equation, this means that if the basic equation changes, it will cause a chain reaction which could probably be the cause as to why an equation proposed by a neural network, is incapable of performing correctly. Neural network is probably not capable of understanding the nonlinearity variable if it does not have enough examples of it.

In a nonlinear model, the nonlinear function is usually regarded as an independent variable which interact with the rest of the linear model, while they cannot be separated, it can be understood that the nonlinearity element is always present in all equations but in different capacities that affect the linearity in different scales.

With this statement in mind, it was observed that based on model predictive control: To obtain the final position, P_f , of an object, it is necessary to consider all the translations that the object will do in a certain period of time. The number of translations needed to obtain the final position are of continuous nature, this means that the number of translations needed will depend entirely on the model we choose and it will not affect the result as long as the trajectories are being represented in a continuous manner.

$$\sum_{i=1}^n P_f = (P_j - P_{j+1}) + (P_{j+1} - P_{j+2}) \dots + (P_{n-1} - P_n) \quad (4.5)$$

In case of the model predictive control for this problem in particular, the translations of the ball are calculated using the equation [2.1], which takes into account all the physical properties that affect the ball and then apply a step-in time, in order to predict the immediate result of the environment and the ball interaction and reaction to each other.

$$V_0 + \int_0^t \dot{V} dt' = -\frac{1}{2m} C_D p A \|V\|V + \frac{1}{2m} C_M p A r \omega \times V + g \quad (4.6)$$

$$P_j = P_0 + \int_0^t V dt' = V_0 + \int_0^t \dot{V} dt' \quad (4.7)$$

While the equation remains the same for each position within the translation of the object, P_j , due to the nonlinear element, $\frac{1}{2m} C_M p A r \omega \times V$, of the equation, each translation will behave differently on each segment, meaning that every translation is unique on its own accord. It is this unique behaviour of the translation segments that can be linked to a neural network feature, which would mean that every translation, is a distinctive feature of a unique trajectory and can be understood as a pattern.

Since the final position of the object, P_f , is already understood to be the summatory of the unique features composed of translations, because of the commutative property, it can then be understood that no matter in which order the translations happen, the result of the final position will remain the same. This indicates that the uniqueness of each translation does not apply to the final position due to the ability of shifting elements and the final product is not affected. This property can be linked to the neural network as a class, meaning that the final position of each trajectory will be shared by several other summatory of trajectories.

Where each of the inputs of the neural network can now be understood as the trajectories of the ball and the activation of the neuron like the class it belongs to.

$$a = \omega_1 x_1 + \omega_2 x_2, + \dots + \omega_n x_n$$
$$x_n = (P_n - P_{n+1}) \quad (4.8)$$
$$a = P_j$$

4.5 Prediction of the ball and expected results

To start generating the prediction of the ball, a real-life scenario is defined to obtain new data that will serve as the database for the machine learning tools. The performance of the tools are analysed, and the results are compared to another similar research if available.

Two examples of a machine learning approach can be seen in Figure [4.5.1] and Figure [4.5.2]

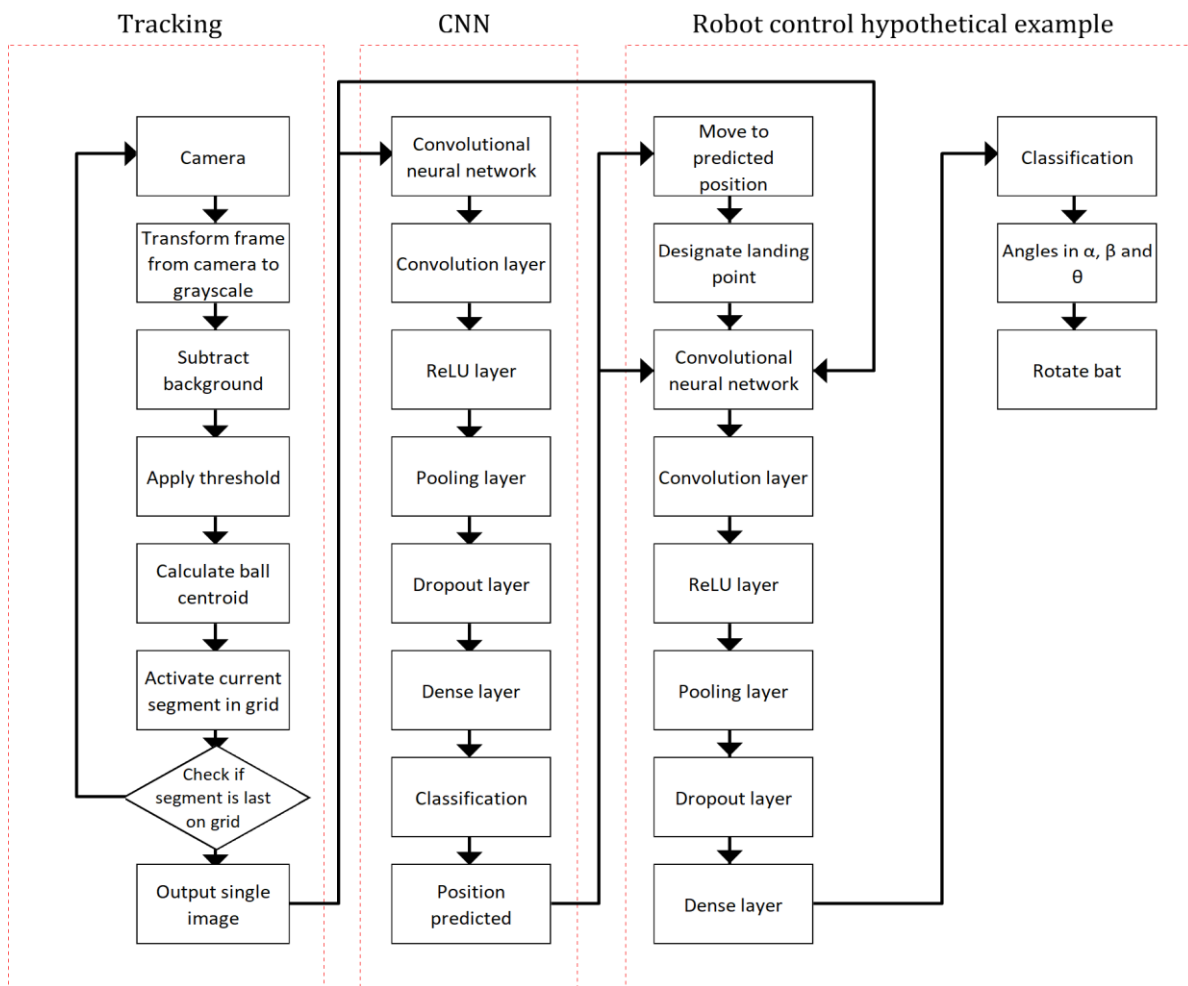


Figure 4.5.1 Proposed Neural Network methodology.

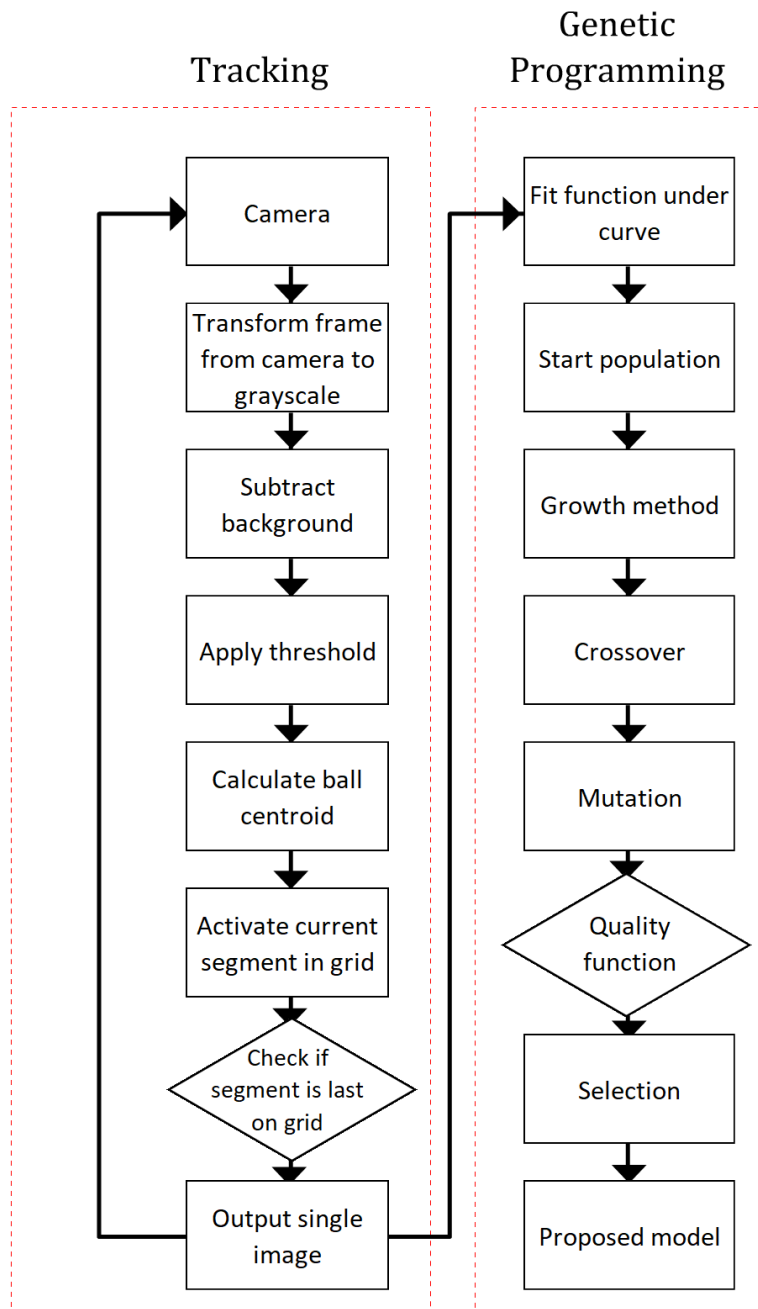


Figure 4.5.2 Proposed Genetic Programming training methodology.



Chapter 5

Experiments and results

This chapter details the results of applying each step of the methodology. Each machine learning result will be discussed and assess the validity of the hypothesis, alongside the novel approach to the nonlinearity within the prediction problem.

5.1 Genetic programming

When applying genetic programming, many examples of the trajectories of the ball were created. Each of these examples where different types of drops, top spin, back spin and bouncing of the ball. This same database is used for the neural network.

The initial testing consisted in finding a replacement for the mathematical model in just the aerodynamic equation, through genetic programming. The rest of the current prediction approach remained unchanged.

This approach has already been attempted very recently but using neural networks instead of genetic programming [86]. The purpose of this test is to compare the results and analyse the response of system when more examples with spin are added to the training.

After proposing an initial example of the trajectory of the ball, the toolbox GPLab from Matlab was used with the following code:

The results obtained were the following. An equation was found to be able to fit the curve of the ball that results from the flight and bouncing on the table.

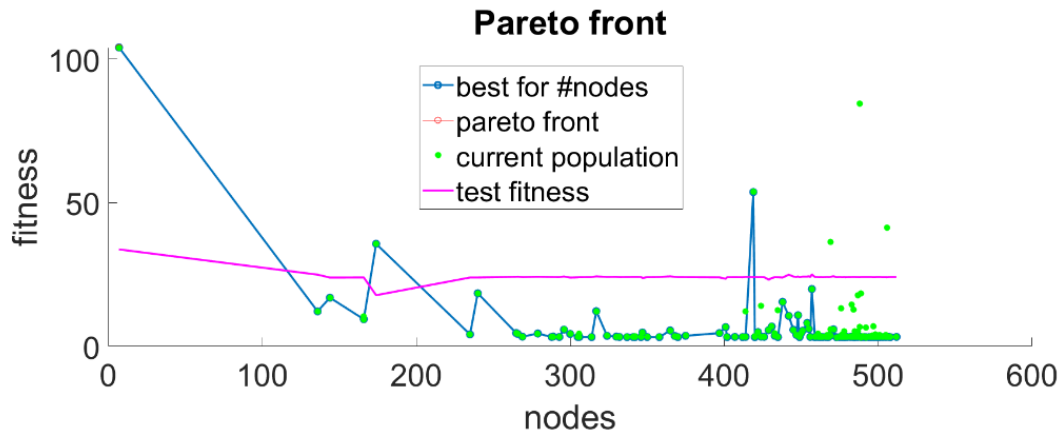


Figure 5.1.2 Pareto front graphic comparing the fitness function to the number of nodes applied and their current population.

The Pareto front shows the relationship between the nodes generation and how they performed when evaluated with the current fitness function. As the population increased, the nodes were able to converge and keep under the fitness required.

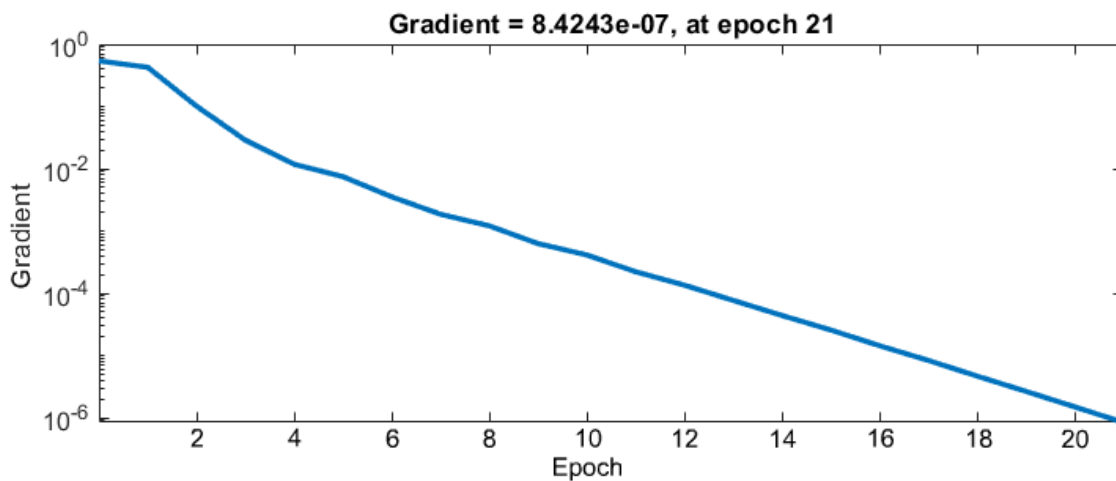


Figure 5.1.3 Graphic representing the gradient of the fitness function.

The gradient of the fitness function was successfully lowering with each epoch, reaching the lowest point at epoch 21, this indicates that the equation was feasible in a relatively short number of generations. The number of nodes does not need to be increased nor the pool size.

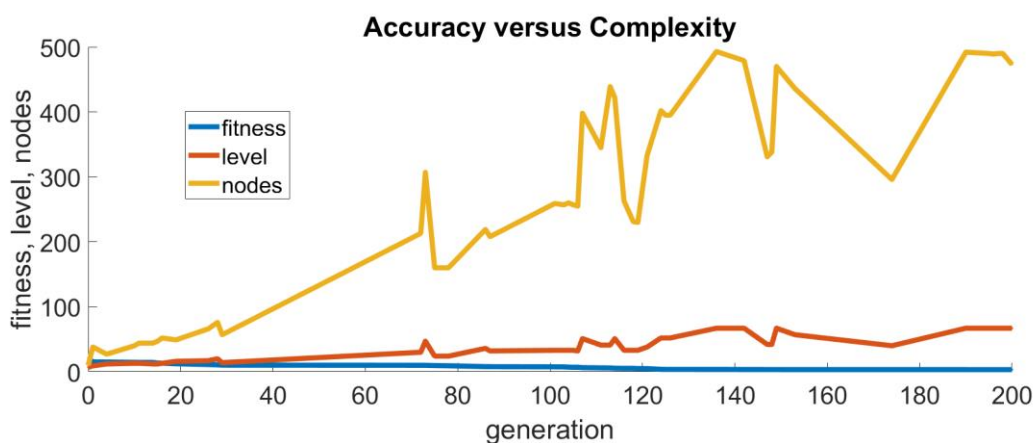



Figure 5.1.4 Graphic comparing the accuracy against the complexity of the fitness level.

When analysing the graphic of accuracy, the level at which the equation was able to fulfil the fitness function always remained low, which means that the growth



of nodes was not entirely necessary, but it did not affect negatively either, which demonstrates that the equation may be highly flexible in nature whenever required.

One important statement to take into consideration is that the more complex the resulting equation is, the more time it will take to solve it, hence causing the prediction to take longer to compute.

To test the accuracy of the resulting equation, several attempts with lower or higher generations and nodes were conducted and the results were as following.

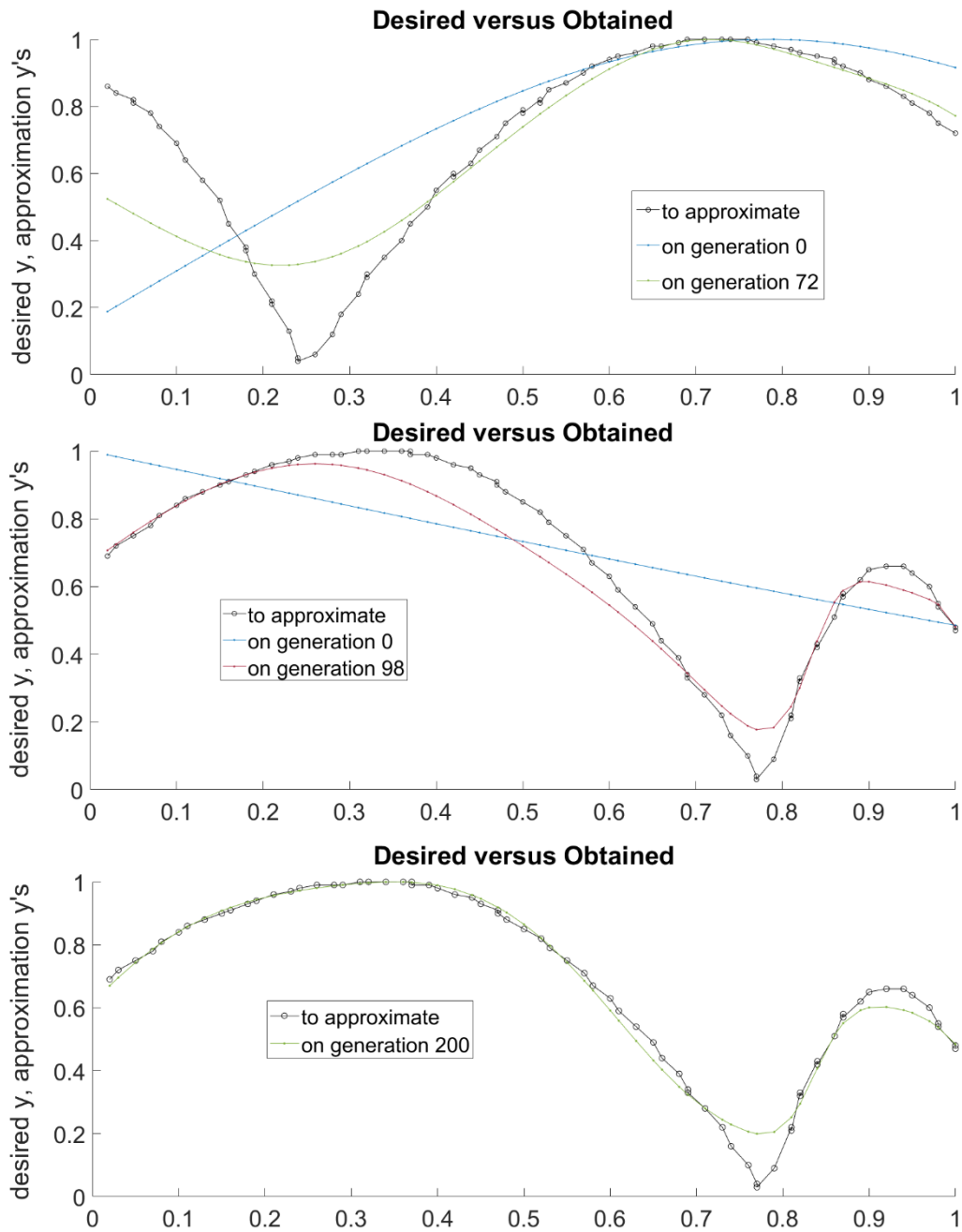



Figure 5.15 Three graphics representing the result of fitting the curve with genetic programming with different number of generations available, 72 (top), 98 (middle) and 200 (bottom).



While some equations were able to be solved with a low number of generations and nodes, like the one shown in Figure [5.1.5], some equations might need more than 200 generations, which might complicate the problem further.

After more testing and careful consideration of the results, it was clear that the more examples were presented to this machine learning approach, the more inconsistent the resulting equation. The complexity for the behaviour of the ball with no spin was manageable, but when spin was introduced and the trajectory increased the complexity dramatically, the new equations were not short enough to let the prediction compute in an acceptable time frame and on some occasions, the accuracy was lowering due to the system believing that all the balls now behave with added spin, which is incorrect.

This last statement is perhaps the most important discovery of this particular approach, ***when using machine learning approach to fit the curve of the trajectory and come up with an equation equivalent to the mathematical model of the aerodynamic model, the spin introduce a nonlinearity in the behaviour that causes the machine learning algorithm to treat it as a general rule for the ball instead of a non-recurring event that might or might not be present at random and it will end up over fitting.***

The only possible way to confirm this hypothesis was using a different machine learning algorithm and test it with a new approach.

5.2 Convolutional Neural Network

The convolutional neural network is trained and classify the trajectory of the ball into twenty different classes per hyper plane. The convolutional neural network was programmed in Matlab, using the Deep Learning Toolbox.

```
load CLASS.mat
load SAMPLE.mat

% Choose a Training Function
% 'trainlm' is usually fastest.
% 'trainbr' takes longer but may be better for challenging problems.
% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainbr'; % Scaled conjugate gradient backpropagation.

% Create a Pattern Recognition Network
hiddenLayerSize = 5;
net = patternnet(hiddenLayerSize, trainFcn);

% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;

% Train the Network
[net,tr] = train(net,x,t);

% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)
tind = vec2ind(t);
yind = vec2ind(y);
percentErrors = sum(tind ~= yind)/numel(tind);

% View the Network
view(net)

% Plots
figure, plotperform(tr)
figure, plottrainstate(tr)
figure, ploterrhist(e)
figure, plotconfusion(t,y)
figure, plotroc(t,y)
```

A combination of different quantities of neurons in the hidden layer was used to test the efficiency of the network and the results were as following.

Confusion Matrix

1	118 4.9%	13 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.1%	1 0.0%	0 0.0%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.5% 14.5%	
2	1 0.0%	107 4.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.2% 1.8%	
3	0 0.0%	0 0.0%	105 4.4%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.1% 1.9%	
4	1 0.0%	0 0.0%	5 2.8%	66 2.8%	3 0.1%	15 0.6%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	3 0.1%	0 0.0%	67.3% 32.7%
5	0 0.0%	0 0.0%	6 0.2%	48 2.0%	111 4.6%	10 0.4%	7 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	60.7% 39.3%
6	0 0.0%	0 0.0%	1 0.0%	0 0.0%	5 0.2%	82 3.4%	4 0.2%	3 0.1%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.2%	1 0.0%	80.4% 19.6%
7	0 0.0%	0 0.0%	1 0.0%	5 0.2%	0 0.0%	11 0.5%	81 3.4%	13 0.5%	2 0.1%	1 0.0%	0 0.0%	3 0.1%	0 0.0%	1 0.0%	6 0.2%	6 0.2%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	61.4% 38.6%
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	19 0.8%	89 3.7%	1 0.0%	5 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	76.7% 23.3%
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.1%	11 0.5%	105 4.4%	8 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	82.7% 17.3%
10	0 0.0%	0 0.0%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	3 0.1%	10 0.4%	98 4.1%	6 0.2%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.3% 19.7%
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	4 0.2%	99 4.1%	23 1.0%	10 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	72.3% 27.7%
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	13 0.5%	66 2.8%	13 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	71.7% 28.3%
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	18 0.8%	82 3.4%	7 0.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.2% 24.8%
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.3%	15 0.6%	101 4.2%	12 0.5%	3 0.1%	1 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	71.6% 28.4%
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 0.4%	94 3.9%	10 0.4%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	82.5% 17.5%
16	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	8 0.3%	86 3.6%	9 0.4%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	81.9% 18.1%
17	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.6%	88 3.7%	4 0.2%	1 0.0%	1 0.0%	0 0.0%	80.7% 19.3%
18	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	18 0.8%	99 4.1%	29 1.2%	5 0.2%	0 0.0%	65.1% 34.9%
19	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	11 0.5%	81 3.4%	11 0.5%	0 0.0%	78.6% 21.4%
20	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	1 0.0%	101 4.2%	97.1% 2.9%
	98.3% 1.7%	89.2% 10.8%	87.5% 12.5%	55.0% 45.0%	92.5% 7.5%	68.3% 31.7%	67.5% 32.5%	74.2% 25.8%	87.5% 12.5%	81.7% 18.3%	82.5% 17.5%	55.0% 45.0%	68.3% 31.7%	84.2% 15.8%	78.3% 21.7%	71.7% 28.3%	73.3% 26.7%	82.5% 17.5%	67.5% 32.5%	84.2% 15.8%	77.5% 22.5%	
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20		

Table 5.2.1 Table showing the result of using two neurons in each hidden layer.

		Confusion Matrix																					
Output Class		Target Class																					
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	Accuracy	Loss
1		114 4.8%	13 0.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	89.8%	10.2%
2		6 0.2%	104 4.3%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	94.5%	5.5%
3		0 0.0%	2 0.1%	112 4.7%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	97.4%	2.6%
4		0 0.0%	0 0.0%	7 0.3%	105 4.4%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	91.3%	8.7%
5		0 0.0%	0 0.0%	0 0.0%	14 0.6%	105 4.4%	2 0.1%	1 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	84.7%	15.3%
6		0 0.0%	0 0.0%	1 0.0%	0 0.0%	8 0.3%	103 4.3%	6 0.2%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.1%	14.9%
7		0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	15 0.6%	98 4.1%	22 0.9%	2 0.1%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	70.0%	30.0%
8		0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	14 0.6%	81 3.4%	10 0.4%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.7%	24.3%
9		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	6 0.2%	97 4.0%	9 0.4%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.1%	14.9%
10		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.3%	8 0.3%	108 4.5%	7 0.3%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	81.8%	18.2%
11		0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	99 4.1%	15 0.6%	5 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	81.1%	18.9%
12		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 0.4%	81 3.4%	5 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.3%	14.7%
13		0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.2%	22 0.9%	103 4.3%	4 0.2%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	75.7%	24.3%
14		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	6 0.2%	99 4.1%	7 0.3%	2 0.1%	0 0.0%	2 0.1%	0 0.0%	0 0.0%	83.9%	16.1%
15		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	16 0.7%	104 4.3%	17 0.7%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	74.8%	25.2%
16		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.2%	6 0.3%	90 3.8%	16 0.7%	3 0.1%	0 0.0%	0 0.0%	76.9%	23.1%
17		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	10 0.4%	86 3.6%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	86.0%	14.0%
18		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	12 0.5%	83 3.5%	13 0.5%	1 0.0%	0 0.0%	74.8%	25.2%
19		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 0.2%	24 1.0%	100 4.2%	5 0.2%	0 0.0%	75.2%	24.8%
20		0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	3 0.1%	7 0.3%	114 4.8%	8.1%	91.9%	8.1%
		95.0%	86.7%	93.3%	87.5%	87.5%	85.8%	81.7%	67.5%	80.8%	90.0%	82.5%	67.5%	85.8%	82.5%	86.7%	75.0%	71.7%	69.2%	83.3%	95.0%	82.8%	17.2%

Table 5.2.2 Table showing the result of using five neurons in each hidden layer.

Confusion Matrix

1	119 5.0%	14 0.6%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	89.5% 10.5%	
2	1 0.0%	105 4.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	99.1% 0.9%	
3	0 0.0%	1 0.0%	114 4.8%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	98.3% 1.7%	
4	0 0.0%	0 0.0%	6 0.2%	105 4.4%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	91.3% 8.7%	
5	0 0.0%	0 0.0%	0 0.0%	14 0.6%	106 4.4%	3 0.1%	1 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.5% 14.5%	
6	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 0.4%	103 4.3%	5 0.2%	3 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.8% 14.2%	
7	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	14 0.6%	98 4.1%	21 0.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	73.1% 26.9%	
8	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.6%	87 3.6%	8 0.3%	2 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	77.7% 22.3%	
9	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	4 0.2%	99 4.1%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	91.7% 8.3%	
10	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.2%	13 0.5%	105 4.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	85.4% 14.6%	
11	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	8 0.3%	102 4.2%	11 0.5%	6 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	80.3% 19.7%	
12	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.6%	96 4.0%	10 0.4%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	79.3% 20.7%	
13	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	13 0.5%	98 4.1%	4 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	83.8% 16.2%	
14	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	1 0.0%	0 0.0%	6 0.2%	101 4.2%	7 0.3%	5 0.2%	0 0.0%	0 0.0%	0 0.0%	83.5% 16.5%	
15	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	15 0.6%	104 4.3%	10 0.4%	1 0.0%	0 0.0%	0 0.0%	80.0% 20.0%	
16	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 0.4%	100 4.2%	23 1.0%	4 0.2%	0 0.0%	73.5% 26.5%	
17	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	5 0.2%	84 3.5%	2 0.1%	0 0.0%	92.3% 7.7%	
18	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	10 0.4%	99 4.1%	19 0.8%	0 0.0%	77.3% 22.7%	
19	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	2 0.1%	14 0.6%	96 4.0%	7 0.3%	80.7% 19.3%	
20	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.0%	5 0.2%	113 4.7%	95.0% 5.0%	
	99.2% 0.8%	87.5% 12.5%	95.0% 5.0%	87.5% 12.5%	88.3% 11.7%	85.8% 14.2%	81.7% 18.3%	72.5% 27.5%	82.5% 17.5%	87.5% 12.5%	85.0% 15.0%	80.0% 20.0%	81.7% 18.3%	84.2% 15.8%	86.7% 13.3%	83.3% 16.7%	70.0% 30.0%	82.5% 17.5%	80.0% 20.0%	94.2% 5.8%	84.8% 15.2%
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	

Table 5.2.3 Table showing the result of using 10 neurons in each hidden layer.

In the Table [5.2.1], it can be appreciated that the number of neurons is insufficient due to a low accuracy. While in Table [5.2.2], the accuracy increased considerably while using a moderate number of neurons and in Table [5.2.3], the accuracy is the best out of the three cases but the number of neurons and the time needed to train the network in comparison to those of the previous test, demonstrate that the increase in accuracy is not worth the increase in time it takes to compute and the resources needed to train it.

With this test in mind, the rest of the training and results reported will be mostly from the second case, Table [5.2.2], due to being the most cost effective of the three cases.

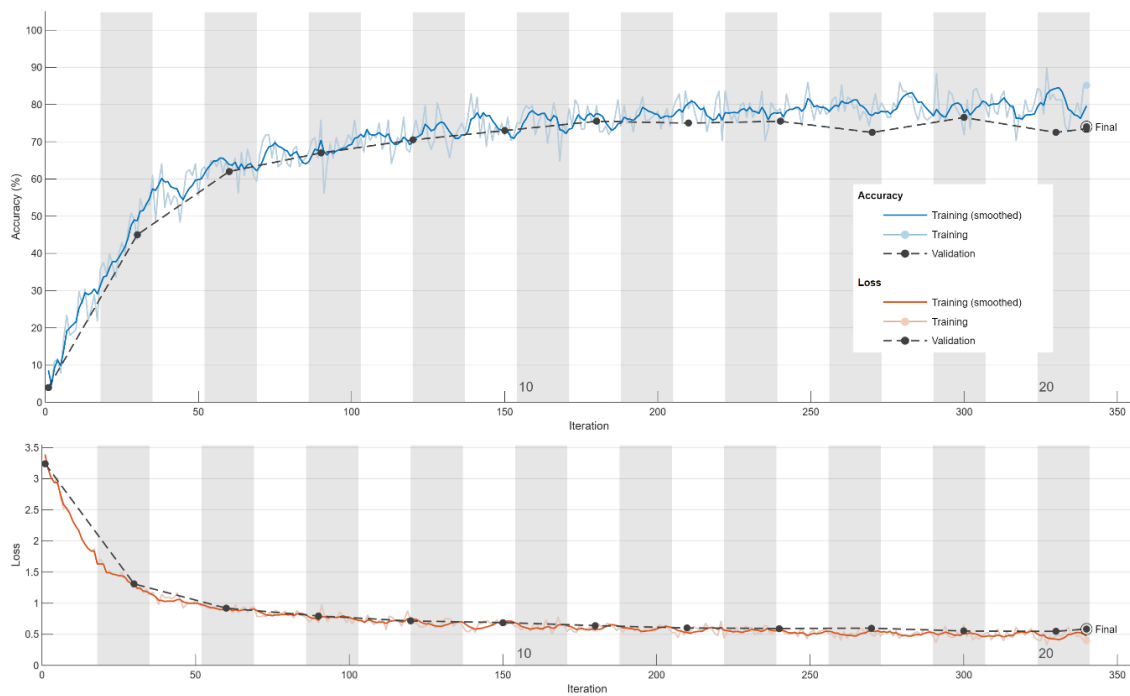


Figure 5.2.1 Graphic representing the accuracy of the convolutional neural network.

The general accuracy of the system is then tested, and the results are appreciated in the Figure [5.2.1]. For this test, the number of iterations was increased to check if there was any case of over fitting, but the data shows that this not the case, which is one of the main issues reported in the previous machine learning approach using genetic programming.

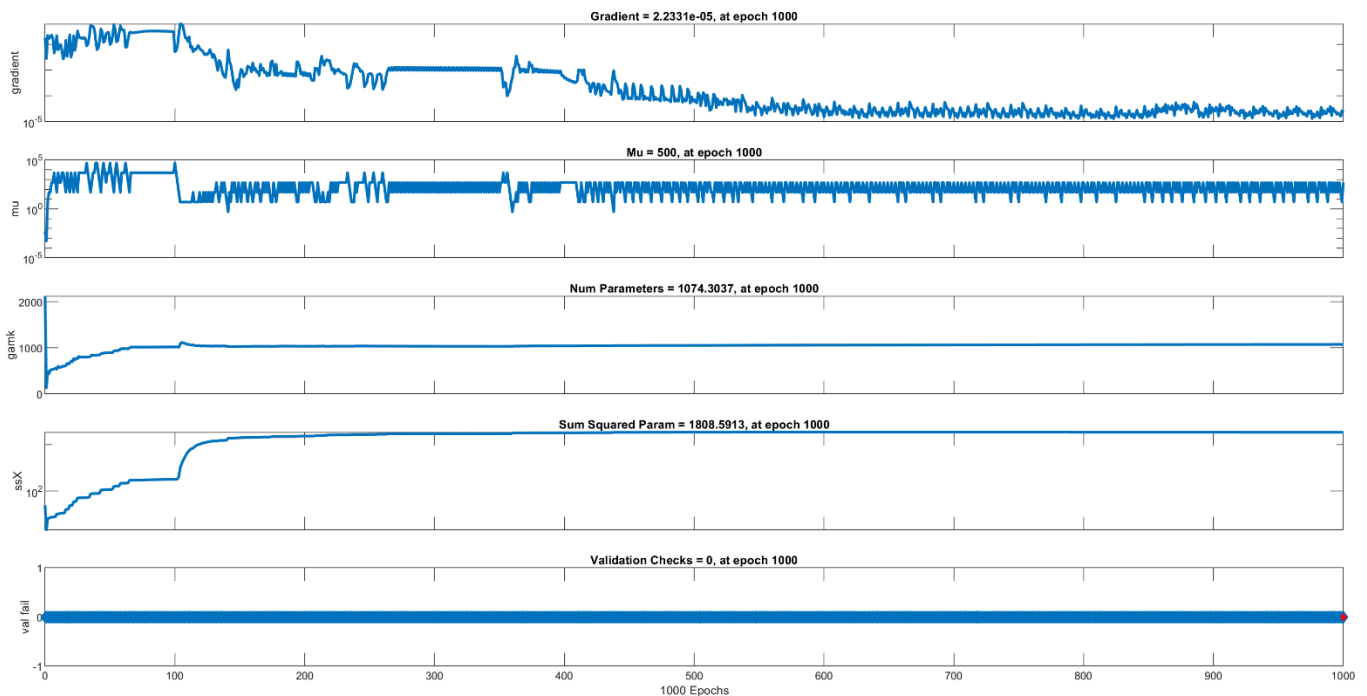


Figure 5.2.2 Graphic representing the gradient results of the convolutional neural network.

When analysing the gradient and the general training state of the network, it can be appreciated that the gradient is convergent, and it reduced to a minimum expression with just half the number of iterations currently being used. This means that the network is capable of learning and classifying the trajectories in an efficient way, but it is important to point out the flat line near the epoch 300. This flat line indicates that the network was probably confusing some of the trajectories but when more data was analysed, it managed to create a vector to successfully separate the

classes as required. This could potentially mean that the network might have a slight weakness towards false local solutions, but it can easily be solved by increasing the number of epochs by a small amount while in training.

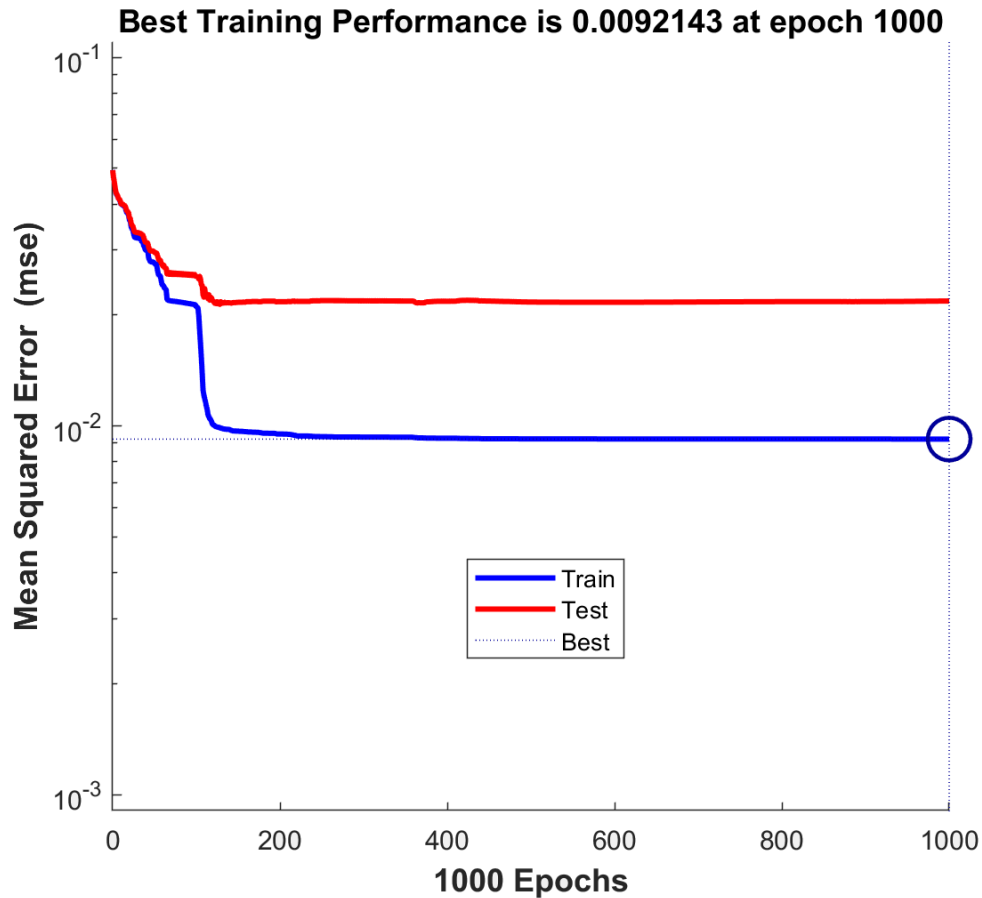


Figure 5.2.3 Graphic representing the training performance of the network.

As mentioned previously, the testing and training of this network has yielded positive results. Even with a high number of epochs, it demonstrated to be a robust approach with no real under fitting and no appreciable over fitting problems.

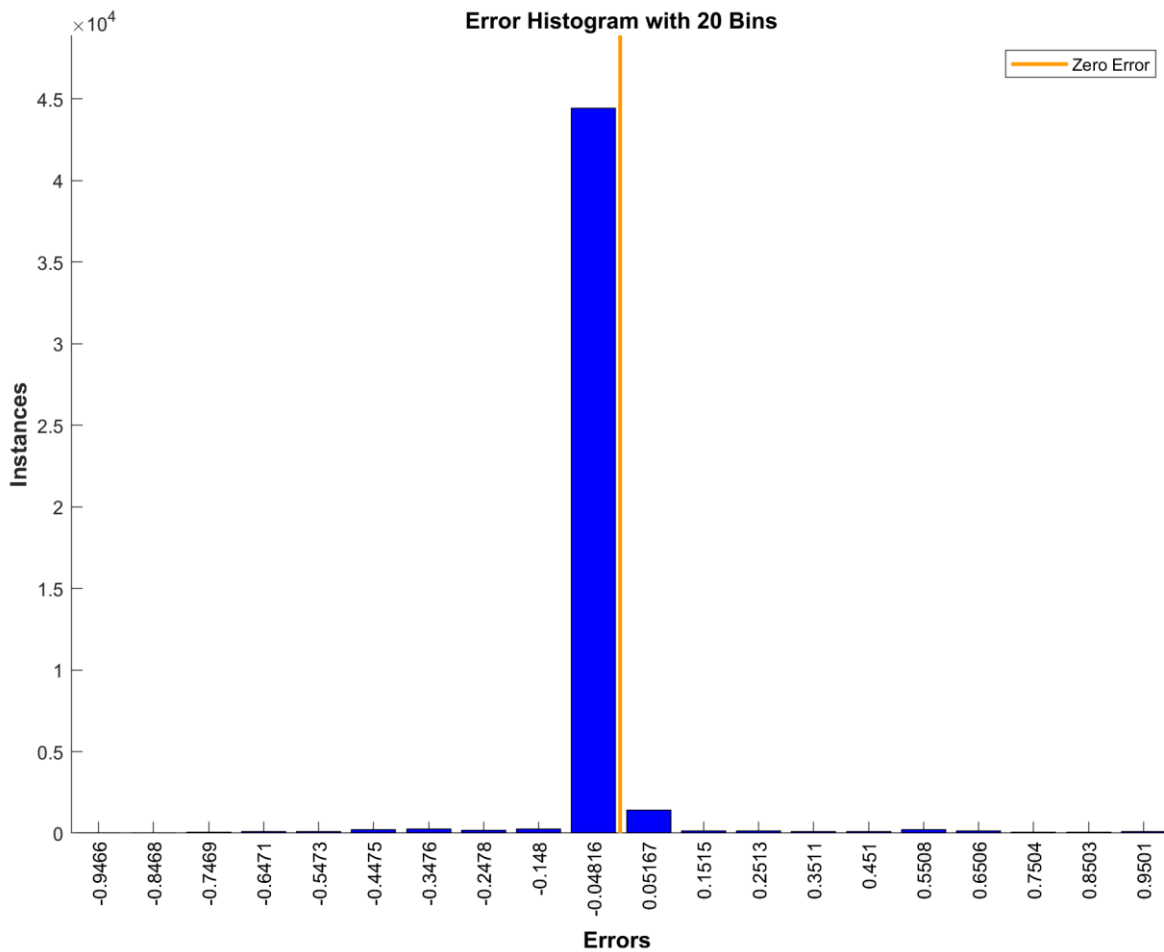


Figure 5.2.4 Graphic representing the error histogram of the network.

Analysing the error histogram, it is appreciated that the error remained mostly constant throughout the training and the testing period, meaning that the dataset was robust enough to be able to deal with new trajectories without much problem and even when there was an error, it did not deviate too much.

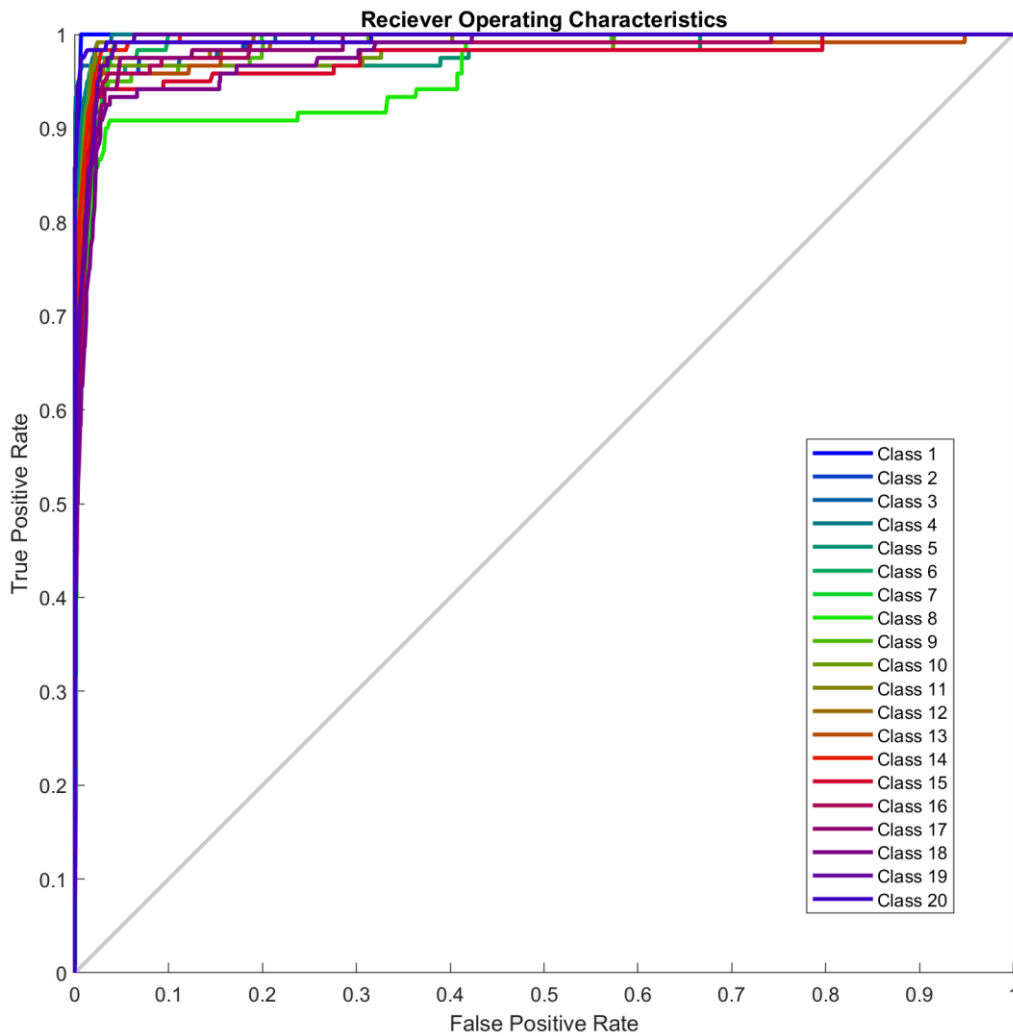



Figure 5.2.5 Graphic representing the positive errors while classifying the inputs.

Figure [5.2.5] shows the number of errors made in each class and the type of error. A true positive imply a trajectory of a different class wrongfully attributed to the class shown, while a false positive imply a trajectory from the class shown wrongfully attributed to another different class. Most of the errors were attributed as true positives for most cases, except for Class 7, the reason for this is not really known but the most probable cause might be a lack of proper training examples



which lead to an under training of that particular Class, this might also explain the flat line that happened in Figure [5.2.3] near the epoch 300.



Chapter 6


Contributions, conclusions and future work

In this chapter the results and personal experiences obtained throughout the research will be contemplated. Emphasize on the successes and errors committed throughout the research will also be mentioned so that the reader takes into consideration some important points that could be used in subsequent research.

6.1 Contributions

This research aimed to fill a gap currently present in the table tennis research. Defined as one of the main objectives, the contribution of this research is to replace the mathematical model with a machine learning approach, which is a problem that is not being properly handled due to the early stages of machine learning as a predictive tool in a nonlinear environment.

The main reason to contribute to this problem is that table tennis is a simple game to understand but, it is really hard to master. So, a problem of this nature is a good prospect to try out different control methodologies and compare the approaches between them. If a problem this simple can be mastered, subsequent problems with a higher level of complexity can now be proposed, so it is of the utmost importance that we are able to tackle this sort of problems first and have a solid basis to construct further research on it.




One of the main contributions of this research was to explain and prove that an alternate methodology can be used not only to replace the mathematical model for an object with nonlinear behaviour, but also as a different approach to other similar machine learning systems. It is this alternate methodology that allows researchers in this problem to analyse, compare and mostly experiment with a different outside perspective their own ideas and hopefully spawn better and more efficient algorithms capable of solving this problem in a more capable manner.

The most important contribution of this research was the approach in terms of defining a new method to link the nonlinear behaviour of a particle into a machine learning system that treats this problem as a classification inspired problem in which the machine learning algorithms excel at. This flexible approach is what probably was needed to master this problem and hopefully inspire a better method in the future.


6.2 Conclusions

Machine learning tools have been used in several areas of knowledge in the last few years, this is due to their flexible approach and great performance solving problems. One of the main advantages of using machine learning is that the algorithms have the ability to learn and adjust themselves in order to find a desired result, so in contrast to more classic approaches that are very precise but rigid in their usage, machine learning allows for a greater liberty in terms of using more or less data, as long as the objective is clear, and the examples used in the data are acceptable enough. The only main disadvantage of most of the tools in machine learning is that they still need to be handled with care because their own flexibility towards big databases can also cause unforeseen problems.



One of these problems that I encountered and probably didn't think of until it happened, was the fact that every single piece of data handled by neural networks is treated equally. While it might sound strange, this can be a very tricky problem that is not easy to understand at first, but it can have a deep impact in the results if not handled correctly. The only way I managed to understand this phenomenon was until it was too late and the previous positive results I managed to obtain through genetic programming, were now all being negative without any visible reason. It was not until I repeated my procedure a couple of times that I understood that treating all data as equally important means that any exception in our database, as in one example that is extraordinary or simply not useful, can corrupt the rest of the training done by all the good examples. What we would naturally understand as an "exception to the rule", neural networks treat it simply as another "average behaviour of the problem to solve", so a big change would be needed to avoid this type of misconceptions.

It was not until I realised the problem with the corrupted data that I had to make a choice, which was work on the algorithm so it can understand that some examples might be exceptions and ignore those to avoid the corruption of the learnt pattern or start from scratch and propose a new approach to treat the data differently. At first, I was going to just add an exclusion to the genetic programming algorithm, but I realised it would be complicated and would render the whole idea of flexibility useless since it would now be limiting the capability of the machine learning approach. Starting from scratch was probably a risky idea but it helped realise that the problem was not on the data, it was on the problem approach itself. While we are taught to approach most problems in one or two ways, those types of methodologies are only useful for classical systems, so something more modern that is built entirely on a different idea is not always compatible with those old systems and need the problem to be redefined in such a way that to avoid the weaknesses of the new method. Coming up with that redefinition of the problem is



the key to properly use machine learning and I cannot stress it enough, the problem is not the algorithm, and it is not always the database, it is the approach to the problem.


After months of being stuck thinking on how to approach the problem, I realised that the only way to solve it was to ignore most methodologies in classic control and instead use those ideas as a basis to come up with a new perspective that is applicable to the problem. It was not easy to make the connection between a system that predicts an object trajectory through serialised equations and moments in time, and a system that is purely for classification purposes. I am convinced this leap of thought will be useful if it is applied to different problems, which might not be easy, but it is worth a try.

6.3 Future work

Confirming that machine learning is capable of predicting the trajectory of a nonlinear object within a strict time limit is the first step towards being able to trust robots with more complex tasks but, in my opinion, there is still much to do in regards of prediction that could potentially be a breakthrough in several disciplines of science and engineering.

One of the main objectives of this research was being able to base a new approach from classical systems and modernise the methodology to see the results. Now that the results prove it is a positive improvement, what could be done after this, is to keep pushing the prediction element even further.


Some research has already been done regarding a machine capable to attempt to emulate the phenomena of instinct. In table tennis, movement is the key



to lower the error between the current and the desired position, which means that implementing some sort of instinct-based algorithm could prove useful in this regard. Since it has already been established that the importance of the solution rely mostly on how the problem is proposed, I am convinced that the next logical step could potentially involve a new approach to be able to predict the opponent movement before it hits the ball, which then will work in conjunction with the current methodology of predicting the ball trajectory through neural networks. This instinct approach could potentially involve deep learning since the dataset used to predict a person movement might be a bit too difficult to process, so instead of deciding which areas might be of interest, the computer itself will have to decide the pattern to follow through.


References


- [1] Y. Zhang, R. Xiong, Y. Zhao, and J. Wang, Real-time spin estimation of ping-pong ball using its natural brand. *IEEE Trans. Instrum. Meas.*, vol. 64, no. 8, pp. 2280–2290, Aug. 2015.
- [2] Y. Zhao, Y. Zhang, R. Xiong, and J. Wang, Optimal state estimation of spinning ping-pong ball using continuous motion model. *IEEE Trans. Instrum. Meas.*, vol. 64, no. 8, pp. 2208–2216, Aug. 2015.
- [3] H. Su, Z. Fang, D. Xu, and M. Tan, Trajectory prediction of spinning ball based on fuzzy filtering and local modeling for robotic ping-pong player. *IEEE Trans. Instrum. Meas.*, vol. 62, no. 11, pp. 2890–2900, Nov. 2013.
- [4] Y. Huang, D. Xu, M. Tan, and H. Su, Trajectory prediction of spinning ball for ping-pong player robot. In *Proceedings. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2011, pp. 3434–3439
- [5] K. P. Modi, F. Sahin and E. Saber, An application of human robot interaction: Development of a ping-pong playing robotic arm. *Proceedings IEEE Int. Conf. Syst. Man Cybern.*, pp. 1831-1836, Oct. 2005.
- [6] L. Sun, J. Liu, Y. Wang, L. Zhou, Q. Yang and S. He, Ball's flight trajectory prediction for table-tennis game by humanoid robot. *Proceedings IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, pp. 2379-2384, Dec. 2009.
- [7] W. Yingshi, S. Lei, L. Jingtai, Y. Qi, Z. Lu and H. Shan, A novel trajectory prediction approach for table-tennis robot based on nonlinear output feedback observer. *Proceedings IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, pp. 1136-1141, Dec. 2010.
- [8] X. Chen, Y. Tian, Q. Huang, W. Zhang, and Z. Yu, Dynamic model-based ball trajectory prediction for a robot ping-pong player. In *Proceedings. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2010, pp. 603–608.
- [9] Li Jing, Chen Xiaopeng, Huang Qiang, Chen Xuechao, Yu Zhangguo, Duo Yingxian, Designation and Control of Landing Points for Competitive Robotic Table Tennis. *International Journal of Advanced Robotic Systems*, Jul 2015, Vol.12(7)
- [10] P. Yang, D. Xu, H. Wang, and Z. Zhang, Control system design for a 5-dof table tennis robot, in *11th Int. Conf. ICARCV*, 2010, pp. 1731-1735.
- [11] Shi, Q.; Li, C.; Wang, C.; Luo, H.; Huang, Q.; Fukuda, T. Design and implementation of an omnidirectional vision system for robot perception. *Mechatronics* 2017, 41, 58–66.

-
- 
- [12] Koç, O.; Maeda, G.; Peters, J. Online optimal trajectory generation for robot table tennis. *Robot. Auton. Syst.* 2018, 105, 121–137.
- [13] Zhang, Y.H.; Wei, W.; Yu, D.; Zhong, C.W. A tracking and predicting scheme for ping pong robot. *J. Zhejiang Univ. Sci. C* 2011, 12, 110–115.
- [14] Li, H.; Wu, H.; Lou, L.; Kühnlenz, K.; Ravn, O. Ping-pong robotics with high-speed vision system. In *Proceedings of the 2012 12th International Conference on Control Automation Robotics & Vision (ICARCV)*, Guangzhou, China, 5–7 December 2012; pp. 106–111.
- [15] Liu, J.; Fang, Z.; Zhang, K.; Tan, M. Improved high-speed vision system for table tennis robot. In *Proceedings of the 2014 IEEE International Conference on Mechatronics and Automation*, Tianjin, China, 3–6 August 2014; pp. 652–657.
- [16] Chen, X.; Huang, Q.; Zhang, W.; Yu, Z.; Li, R.; Lv, P. Ping-pong trajectory perception and prediction by a PC based High speed four-camera vision system. In *Proceedings of the 2011 9th World Congress on Intelligent Control and Automation*, Taipei, Taiwan, 21–25 June 2011; pp. 1087–1092.
- [17] Yang, P.; Xu, D.; Zhang, Z.; Chen, G.; Tan, M. A vision system with multiple cameras designed for humanoid robots to play table tennis. In *Proceedings of the 2011 IEEE International Conference on Automation Science and Engineering*, Trieste, Italy, 24–27 August 2011; pp. 737–742.
- [18] Liu, Y.; Liu, L. Accurate real-time ball trajectory estimation with onboard stereo camera system for humanoid ping-pong robot. *Robot. Auton. Syst.* 2018, 101, 34–44.
- [19] Lampert, C.H.; Peters, J. Real-time detection of colored objects in multiple camera streams with off-the-shelf hardware components. *J. Real-Time Image Process.* 2012, 7, 31–41.
- [20] Schauwecker, K. Real-Time Stereo Vision on FPGAs with SceneScan. In *Forum Bildverarbeitung 2018*; KIT Scientific Publishing: Deutschland, Germany, 2018; p. 339.
- [21] Zhang, Y.; Zhao, Y.; Xiong, R.; Wang, Y.; Wang, J.; Chu, J. Spin observation and trajectory prediction of a ping-pong ball. In *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA)*, Hong Kong, China, 31 May–7 June 2014; pp. 4108–4114. *Sensors* 2020, 20, 333 23 of 23
- [22] Zhao, Y.; Zhang, Y.; Xiong, R.; Wang, J. Optimal state estimation of spinning ping-pong ball using continuous motion model. *IEEE Trans. Instrum. Meas.* 2015, 64, 2208–2216.


-
- 
- [23] Zhang, Z.; Xu, D.; Tan, M. Visual measurement and prediction of ball trajectory for table tennis robot. *IEEE Trans. Instrum. Meas.* 2010, 59, 3195–3205.
- [24] Wang, P.; Zhang, Q.; Jin, Y.; Ru, F. Studies and simulations on the flight trajectories of spinning table tennis ball via high-speed camera vision tracking system. *Proceedings Inst. Mech. Eng. Part P J. Sports Eng. Technol.* 2019, 233, 210–226.
- [25] Huang, Y.; Xu, D.; Tan, M.; Su, H. Trajectory prediction of spinning ball for ping-pong player robot. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, 25–30 September 2011; pp. 3434–3439.
- [26] Zhao, Y.; Xiong, R.; Zhang, Y. Rebound modeling of spinning ping-pong ball based on multiple visual measurements. *IEEE Trans. Instrum. Meas.* 2016, 65, 1836–1846.
- [27] Su, H.; Fang, Z.; Xu, D.; Tan, M. Trajectory prediction of spinning ball based on fuzzy filtering and local modeling for robotic ping-pong player. *IEEE Trans. Instrum. Meas.* 2013, 62, 2890–2900.
- [28] Zhang, Y.; Xiong, R.; Zhao, Y.; Wang, J. Real-time spin estimation of ping-pong ball using its natural brand. *IEEE Trans. Instrum. Meas.* 2015, 64, 2280–2290.
- [29] Wang, Q.; Zhang, K.; Wang, D. The trajectory prediction and analysis of spinning ball for a table tennis robot application. In *Proceedings of the 4th Annual IEEE International Conference on Cyber Technology in Automation, Control and Intelligent*, Hong Kong, China, 4–7 June 2014; pp. 496–501.
- [30] Nakashima, A.; Ogawa, Y.; Kobayashi, Y.; Hayakawa, Y. Modeling of rebound phenomenon of a rigid ball with friction and elastic effects. In *Proceedings of the 2010 American Control Conference*, Baltimore, MD, USA, 30 June–2 July 2010; pp. 1410–1415.
- [31] Bao, H.; Chen, X.; Wang, Z.T.; Pan, M.; Meng, F. Bouncing model for the table tennis trajectory prediction and the strategy of hitting the ball. In *Proceedings of the 2012 IEEE International Conference on Mechatronics and Automation*, Chengdu, China, 5–8 August 2012; pp. 2002–2006.
- [32] Matsushima, M.; Hashimoto, T.; Takeuchi, M.; Miyazaki, F. A learning approach to robotic table tennis. *IEEE Trans. Robot.* 2005, 21, 767–771.
- [33] Miyazaki, F.; Matsushima, M.; Takeuchi, M. Learning to dynamically manipulate: A table tennis robot controls a ball and rallies with a human being.
-

-
- In *Advances in Robot Control*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 317–341.
- [34] Zhao, Y.; Xiong, R.; Zhang, Y. Model based motion state estimation and trajectory prediction of spinning ball for ping-pong robots using expectation-maximization algorithm. *J. Intell. Robot. Syst.* 2017, 87, 407–423.
 - [35] Deng, Z.; Cheng, X.; Ikenaga, T. Ball-like observation model and multi-peak distribution estimation-based particle filter for 3D Ping-pong ball tracking. In *Proceedings of the 2017 Fifteenth IAPR International Conference on Machine Vision Applications (MVA)*, Nagoya, Japan, 8–12 May 2017; pp. 390–393.
 - [36] Payeur, P.; Le-Huy, H.; Gosselin, C.M. Trajectory prediction for moving objects using artificial neural networks. *IEEE Trans. Ind. Electron.* 1995, 42, 147–158.
 - [37] Nakashima, A.; Takayanagi, K.; Hayakawa, Y. A learning method for returning ball in robotic table tennis. In *Proceedings of the 2014 International Conference on Multisensor Fusion and Information Integration for Intelligent Systems (MFI)*, Beijing, China, 28–29 September 2014; pp. 1–6.
 - [38] Zhang, Z. Determining the epipolar geometry and its uncertainty: A review. *Int. J. Comput. Vis.* 1998,
 - [39] Biagiotti, L.; Melchiorri, C. *Trajectory Planning for Automatic Machines and Robots*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2008.
 - [40] Riad Akrou, Abbas Abdolmaleki, Hany Abdulsamad, Jan Peters, and Gerhard Neumann. Model-free trajectory-based policy optimization with monotonic improvement. *J. Mach. Learn. Res.*, 2016.
 - [41] Russell Anderson. *A Robot Ping-Pong Player: Experiments in Real-Time Intelligent Control*. MIT Press, 1988.
 - [42] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pages 41–48, 2009.
 - [43] John Billingsley. *Robot ping pong*. Practical Computing, 1983.
 - [44] Krzysztof Choromanski et al. Structured evolution with compact architectures for scalable policy optimization. In *ICML*, 2018.
 - [45] Krzysztof Choromanski et al. From complexity to simplicity: Adaptive es-active subspaces for blackbox optimization. In *NeurIPS*, 2019.
 - [46] Krzysztof Choromanski et al. Provably robust blackbox optimization for reinforcement learning. In *CoRL*, 2019.

-
- 
- [47] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. <http://pybullet.org>, 2016–2019.
- [48] Chelsea Finn and Sergey Levine. Meta-learning and universality: Deep representations and gradient descent can approximate any learning algorithm. In ICLR, 2018.
- [49] Yapeng Gao, Jonas Tebbe, Julian Krismer, and Andreas Zell. Markerless racket pose detection and stroke classification based on stereovision for table tennis robots. IEEE Robotic Computing, 2019.
- [50] J. Hartley. Toshiba progress towards sensory control in real time. The Industrial Robot 14-1, pages 50–52, 1983.
- [51] Hideaki Hashimoto, Fumio Ozaki, and Kuniji Osuka. Development of ping-pong robot system using 7 degree of freedom direct drive robots. In Industrial Applications of Robotics and Machine Vision, 1987.
- [52] Yanlong Huang, Bernhard Schölkopf, and Jan Peters. Learning optimal striking points for a ping-pong playing robot. IROS, 2015.
- [53] Yanlong Huang, Dieter Buchler, Okan Koç, Bernhard Schölkopf, and Jan Peters. Jointly learning trajectory generation and hitting point prediction in robot table tennis. IEEE-RAS Humanoids, 2016.
- [54] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. ICRA, 2002.
- [55] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In CVPR, 2019.
- [56] Dmitry Kalashnikov et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. CoRL, 2018.
- [57] John Knight and David Lowery. Pingpong-playing robot controlled by a microcomputer. Microprocessors and Microsystems – Embedded Hardware Design, 1986.
- [58] Okan Koç, Guilherme Maeda, and Jan Peters. Online optimal trajectory generation for robot table tennis. Robotics & Autonomous Systems, 2018.
- [59] Timothy Lillicrap, Jonathan Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. ICLR, 2016.

-
- 
- [60] Reza Mahjourian, Navdeep Jaitly, Nevena Lazic, Sergey Levine, and Risto Miikkulainen. Hierarchical policy design for sample-efficient learning of robot table tennis through self-play. arXiv:1811.12927, 2018.
- [61] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search provides a competitive approach to reinforcement learning. NeurIPS, 2018.
- [62] Michiya Matsushima, Takaaki Hashimoto, and Fumio Miyazaki. Learning to the robot table tennis task-ball control and rally with a human. IEEE International Conference on Systems, Man and Cybernetics, 2003.
- [63] Michiya Matsushima, Takaaki Hashimoto, Masahiro Takeuchi, and Fumio Miyazaki. A learning approach to robotic table tennis. IEEE Transactions on Robotics, 2005.
- [64] Fumio Miyazaki, Masahiro Takeuchi, Michiya Matsushima, Takamichi Kusano, and Takaaki Hashimoto. Realization of the table tennis task based on virtual targets. ICRA, 2002.
- [65] Katharina Muelling and Jan Peters. A computational model of human table tennis for robot application. In AMS, 2009.
- [66] Katharina Muelling, Jens Kober, and Jan Peters. A biomimetic approach to robot table tennis. Adaptive Behavior, 2010.
- [67] Katharina Muelling, Jens Kober, and Jan Peters. Learning table tennis with a mixture of motor primitives. IEEE-RAS Humanoids, 2010.
- [68] Katharina Muelling, Jens Kober, and Jan Peters. Simulating human table tennis with a biomimetic robot setup. In Simulation of Adaptive Behavior, 2010.
- [69] Katharina Muelling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. The International Journal of Robotics Research, 2012.
- [70] Anusha Nagabandi et al. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In ICRA, 2018.
- [71] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. FoCM, 2017.
- [72] Aravind Rajeswaran, Kendall Lowrey, Emanuel Todorov, and Sham M. Kakade. Towards generalization and simplicity in continuous control. In NeurIPS, 2017.
- [73] Marie-Marin Ramanantsoa and Alain Durey. Towards a stroke construction model. The International Journal of Table Tennis Sciences, 1994.
- [74] Tim Salimans et al. Evolution strategies as a scalable alternative to reinforcement learning. arXiv:1703.03864, 2017.

-
- 
- [75] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. arXiv:1707.06347, 2017.
- [76] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. ES-MAML: simple hessian-free meta learning. In ICLR, 2020.
- [77] Yichao Sun, Rong Xiong, Qiuguo Zhu, Jingjing Wu, and Jian Chu. Balance motion generation for a humanoid robot playing table tennis. IEEE-RAS Humanoids, 2011.
- [78] Richard Sutton and Andrew Barto. Reinforcement Learning: An introduction. MIT Press, MA, 1 edition, 1998.
- [79] Richard Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In NeurIPS, 2000.
- [80] Jonas Tebbe, Yapeng Gao, Marc Sastre-Rienietz, and Andreas Zell. A table tennis robot system using an industrial kuka robot arm. GCPR, 2018.
- [81] Josh Tobin et al. Domain randomization and generative models for robotic grasping. In IROS, 2018.
- [82] Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. In NeurIPS, 2016.
- [83] Daan Wierstra, Tom Schaul, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. In 2008 IEEE Congress on Evolutionary Computation, 2008.
- [84] Ronald Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Machine Learning, 1992.
- [85] Yifeng Zhu, Yongsheng Zhao, Lisen Jin, Jingjing Wu, and Rong Xiong. Towards high level skill learning: Learn to return table tennis ball using monte-carlo based policy gradient method. IEEE International Conference on Real-time Computing and Robotics, 2018.
- [86] Wenbo Gao, Laura Graesser, Krzysztof Choromanski, Xingyou Song, Nevena Lazic, Pannag Sanketi, Vikas Sindhwani and Navdeep Jaitly. Robotic Table Tennis with Model-Free Reinforcement Learning. diarXiv: 2003.14398v2 [cs.LG], Columbia University, 27 May 2020
- [87] Hsien-I Lin, Zhangguo Yu, Yi-Chen Huang. Ball Tracking and Trajectory Prediction for Table-Tennis Robots. 2019 9th International Conference on Information Science and Technology (ICIST), pp. 222-227, 2019

-
- 
- [88] M. Heimbach, K. Ebadi and S. Wood, "Improving Object Tracking Accuracy in Video Sequences Subject to Noise and Occlusion Impediments by Combining Feature Tracking with Kalman Filtering," 2018 52nd Asilomar Conference on Signals, Systems, and Computers, 2018
- [89] B. J. Koskovich, M. Rahnemoonfai and M. Starek, Virtualot - A framework enabling real-time coordinate transformation & occlusion sensitive tracking using UAS products deep learning object detection & traditional object tracking techniques, IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium, pp. 6416-6419, 2018.
- [90] Rob Pegoraro, <https://www.flickr.com/photos/robpegoraro/39611490852>, January 2018