

DM

# Real-Time Statistics for Padel Tennis Using Artificial Intelligence

MASTER DISSERTATION

**João Rúben Moniz Pontes**

MASTER IN MATHEMATICS, STATISTICS AND APPLICATIONS



UNIVERSIDADE da MADEIRA

*A Nossa Universidade*

[www.uma.pt](http://www.uma.pt)

February | 2024

# Real-Time Statistics for Padel Tennis Using Artificial Intelligence

MASTER DISSERTATION

**João Rúben Moniz Pontes**

MASTER IN MATHEMATICS, STATISTICS AND APPLICATIONS

SUPERVISOR

Paulo Sérgio Abreu Freitas

## Resumo

O Padel, desporto conhecido pelo seu crescimento explosivo e jogabilidade emocionante, está à beira de uma revolução tecnológica. Com o objetivo de transformar o jogo de Padel através do uso criativo de técnicas de detecção de objetos e Deep Learning, esta dissertação de mestrado investiga a junção da Inteligência Artificial (IA) e do Padel. O principal objetivo é usar a IA para produzir estatísticas em tempo real que darão aos jogadores, treinadores e fãs um melhor conhecimento das complexidades do Padel e dos meios para levar o jogo a novos patamares.

Esta dissertação explora a monitorização e localização em tempo real dos jogadores e da bola dentro do campo, através de algoritmos de visão computacional. As Redes Neurais de Convolução (RNC), um tipo de modelo de Deep Learning, são essenciais para o reconhecimento preciso de eventos e ações importantes durante o jogo.

A criação de um sistema baseado em IA que produz dados instantâneos para partidas de Padel é a inovação central desta dissertação. Estas estatísticas oferecem uma visão analítica e detalhada de cada jogo, tendo em consideração os movimentos dos jogadores, as trajetórias da bola e a dinâmica do jogo. Esta dissertação não promove apenas o Padel, mas também cria novas oportunidades para a utilização de IA em outros desportos.

Palavras Chave: Inteligência Artificial, Redes Neurais de Convolução, Padel

# Abstract

The sport of Padel, known for its explosive growth and exciting gameplay, is on the verge of a technological revolution. With the goal of transforming the game of Padel through the creative use of object detection and deep learning techniques, this master's thesis investigates the junction of Artificial Intelligence (AI) and Padel. The main goal is to use AI to produce real-time statistics that will give players, coaches and fans a better knowledge of the complexities of Padel and the means to take the game to new heights.

This dissertation explores the real-time tracking and localization of players and the ball within the court by utilizing cutting-edge computer vision algorithms. Convolution Neural Networks (CNN), one type of deep learning model, are essential for the precise recognition of important gaming events and actions.

The creation of an AI-driven system that produces in-the-moment data for Padel matches is the central innovation of this dissertation. These statistics offer a detailed and analytical view of each game by taking into account player movements, ball trajectories, and game dynamics. This dissertation not only advances the sport of Padel but also creates new opportunities for the use of AI in other sports analytics.

Key Words: Artificial Intelligence, Convolution Neural Networks, Padel

## Acknowledgments

I want to extend my sincere gratitude to everyone who helped this master's thesis be completed successfully. This study project has been a life-changing adventure made possible by the help, direction, and inspiration of various people and organizations.

First and foremost, I would like to express my sincere gratitude to my thesis adviser, professor Paulo Freitas, whose unshakable commitment and knowledge have greatly influenced this work.

I would like to express my gratitude to the University of Madeira's faculty and professors, whose lectures, helpful criticism, and academic direction have expanded my horizons and honed my research abilities. The university's intellectual setting has served as a consistent source of inspiration. I appreciate the moral support and thought-provoking conversations my coworkers and friends have given me along this trip. Your friendship has helped you overcome obstacles and enjoy your accomplishments more.

I would like to thank Eliano Marques and Quinta do Padel for their support in making this research feasible. Your commitment to academic endeavors has a significant effect on the development of knowledge.

My profound gratitude also extends to my family and girlfriend for their everlasting support and unceasing encouragement. My biggest inspiration has been your love and support.

# Contents

<b>1 Introduction</b>	<b>10</b>
1.1 Motivation	10
1.2 Structure and Organization	11
1.3 Sport of Padel	11
1.3.1 History and Evolution	11
1.3.2 Rules and Gameplay	15
1.4 AI	19
1.4.1 History and Evolution	19
1.4.2 Types	20
1.4.3 Techniques and Algorithms	21
1.4.4 Data and AI	23
1.5 Questions and Research Goals	24
<b>2 State of the art</b>	<b>25</b>
2.1 CNN	25
2.2 YOLO Object Detection Model	41
2.3 Transfer Learning	49
2.4 Pose Estimation	52
2.5 Court Detection	55
2.6 Ball Detection	60
<b>3 Research Methodology</b>	<b>63</b>
3.1 Research Design	63
3.2 Methodology	64
3.2.1 Movement Classification	64
3.2.2 Padel Court Identification	69
3.2.3 Ball Position and Trajectory	75
3.2.4 Components Integration	76
3.2.5 Statistics	77
<b>4 Results Analysis</b>	<b>79</b>
4.1 Movements Classification Model	79
4.1.1 Metrics	79
4.1.2 Results	80
4.1.3 Discussion	81

<b>4.2 Discussion</b> . . . . .	81
<b>4.2.1 Research Limitations and Future Works</b> . . . . .	81
<b>5 Conclusion</b>	83
<b>References</b>	85

# List of Figures

1.1	Jeu de Paume on XVI century [Ferreira, 2004].	12
1.2	Recent padel [RoyalNorwich, 2022].	14
1.3	Padel court dimensions [PadelLab, 2023].	16
1.4	AI components.	22
2.1	CNN Feature Map.	27
2.2	CNN Architecture.	32
2.3	YOLO object detection models timeline [LearnOpenCV, 2023].	42
2.4	YOLO prediction example [Terven and Cordova-Esparza, 2023].	43
2.5	YOLOv3 Multi-scale detection architecture [Terven and Cordova-Esparza, 2023].	45
2.6	a) IoC Formula; b) 3 examples of IoC values for different box predictions [Terven and Cordova-Esparza, 2023].	49
2.7	Transfer Learning between tasks [NLP, 2019].	50
2.8	Keypoints of the skeleton [in OpenCV using OpenPose, 2018].	54
2.9	Open Pose skeleton detection example [in OpenCV using OpenPose, 2018].	55
2.10	Canny Edge application example.	57
2.11	KCF occlusion example [Mehmood et al., 2021].	61
3.1	Methodology procedures' architecture.	64
3.2	Cameras distribution in Quinta do Padel's court.	65
3.3	Example of application of Pose Estimation to Quinta do Padel court game.	66
3.4	Example of a CVAT annotation of a Forehand movement.	68
3.5	Movement classifier structure.	68
3.6	Court boundaries identification structure.	70
3.7	Frame of the empty court.	70
3.8	Frame of the court after applying the canny edge detector.	71
3.9	Dilated Result of the canny edge detection.	72
3.10	Detected court lines.	72
3.11	Detected court horizontal lines.	73
3.12	The extracted frame's four points are shown in the left image, while the reference court image and corresponding points are shown in the right image.	75
3.13	AI System's flow representation.	76
4.1	YOLO confusion matrix.	80



## List of Tables

<a href="#">2.1 Non-linear activation functions.</a> . . . . .	29
<a href="#">3.1 Example of a failed service executed by player 2 in the frame logs table.</a> . . .	78

# 1 Introduction

Padel, a sport that combines squash and tennis, has seen a spectacular rise in popularity in recent years. Padel, which had its beginnings in Mexico in the late 1960s, has grown into a global sensation with millions of fans, a professional tournament circuit, and a network of courts that is quickly developing. Padel is unique not just because it is accessible and open to anyone, but also because it has a wealth of technology integration possibilities, notably in the area of AI. Like many sports, Padel is a complex dynamic activity that produces a ton of data while being played.

AI offers a variety of chances to take Padel to new heights, from real-time player tracking and performance enhancement to automated game analysis and fan involvement. This is possible through machine learning (ML) approaches, using data collected in a controlled environment. The challenge lies in identifying key moments of the game, for example: how to detect in real-time when the ball hits the net and scores a point for team A? How to identify the different types of shots during a point in order to create statistics related to the type of shot per player with the highest point conversion? By employing approaches that enable the identification of these important game moments, through rules or ML models that identify the court boundaries, the ball and player coordinates, it is possible to integrate all these components and create a set of policies that allows the generation of real-time statistics based on them. In this context, AI is this system that identifies key parts of the game, understand its meaning and consequently is capable of generating real-time game statistics by itself. With AI technology's further development, important insights might be unlocked, player abilities could be improved, and spectators could enjoy a richer, more immersive experience.

Padel and AI together offer not only a technological improvement but also a conceptual shift in how people view and interact with sports.

## 1.1 Motivation

This master's thesis was inspired by the fusion of two powerful forces: the passionate and vibrant world of Padel and the revolutionary potential of AI and its techniques. While AI has been incorporated into a number of industries, including autonomous driving and healthcare, Padel remains one of the sports where its potential has not yet been fully realized. The motivation for this study stems from the desire to create a world where real-time generation of statistics provides players with a better understanding of their performance during the game, enabling them to take actions to improve highlighted aspects based on these statistics.

## 1.2 Structure and Organization

The dissertation is organized into five chapters. Chapter 1 presents a comprehensive literature review on the integration of AI and provides an introduction to Padel sports and outlines the objectives of the study. Chapter 2 outlines relevant applications for the dissertation development. Chapter 3 discusses the methodologies employed in this research, including data collection techniques and AI model implementations. Chapter 4 presents the results and findings obtained from the analysis of AI applications in Padel sports. Finally, Chapter 5 concludes the thesis with a summary of the key insights and proposes future directions for research.

## 1.3 Sport of Padel

Padel, the sport that arouses passions all over the world, continues to grow in popularity worldwide and is one of the fastest-growing sports in terms of numbers playing. Its unique characteristics and gameplay mechanics present a lot of interest as a participation sport and, according to the International Padel Federation (FIP), there are currently about 25 million players worldwide, with nations like Mexico, Argentina, and Spain seeing huge popularity among amateur and professional players.

This chapter explores the sports of Padel, with a particular focus on its history, rules and popularity.

### 1.3.1 History and Evolution

Although Padel is a relatively new activity, its roots can be found in the racket sports with the longest histories, like tennis or badminton. Two stories exist regarding the origin of the Padel. The first edition discusses the history of Padel with other racket sports like badminton or tennis, as well as its shared French Jeu de Paume ancestry. Later, two games named Paddle Tennis and Platform Tennis began to be played in the United States [González-Carvajal, 2006]. These are scaled-down versions of tennis, but they share a lot in common with Padel. According to the International Padel Federation's recognized second version, the sport was developed in Mexico in 1969 by Enrique Corcuera.

### Historical Background

The sport of Padel tennis is a member of the ball and Padel game family. Although its modern origins may be traced to the turn of the 20th century, its earliest precedents may be found in the "Jeu du Paume" around the end of the 13th century, illustrated in Figure 1.1 [González-Carvajal, 2006].

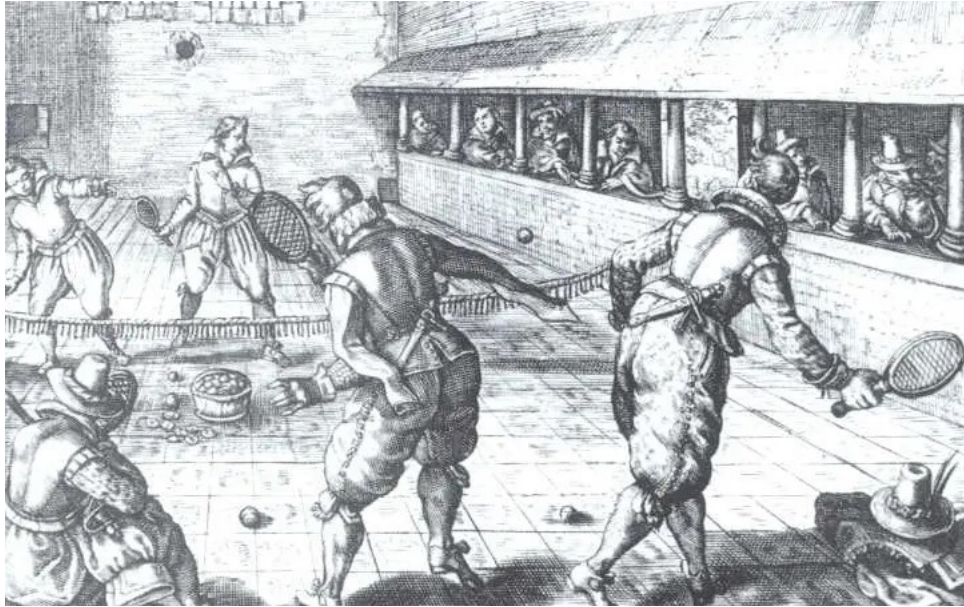


Figure 1.1: Jeu de Paume on XVI century [Ferreira, 2004](#).

This practice, which has its own rules, laws, and statutes, first appeared in France and distinguished between two sorts of games based on whether the ball was struck with bats and sticks, pushed with small instruments, or thrown by hand. The second variety is more widely played since it requires less room for play and allows for longer game times because it is simpler to maintain the ball in play without it touching the ground [Gillmeister, 2008](#), [Almendros, 2009](#).

The "lounge paume" (long palm), which was played outdoors, and the "courte paume" (short palm), which was played in enclosed areas and is more akin to the Padel of today, were the two modalities used in the game, which was played by three or more opponents on each side [Gillmeister, 2008](#).

In the 19th century, different sources affirm the existence of a sport similar to Padel in the basements of English ships, using oars to hit the ball and allowing bouncing off the walls of the ship due to the limited playing surface. This game achieved great popularity as a pastime for sailors and passengers [Gillmeister, 2008](#), [Almendros, 2009](#).

## **Paddle Tennis**

Later, in 1898, the American Reverend Frank Beal of Albion (Michigan) modified a tennis court to teach play to young children by halving its size, replacing the ball with a different foam rubber one, and replacing the racket with a wooden paddle similar to those they used to play with on the beach [Almonacid-Cruz, 2011](#).

This method attained great popularity in Michigan. It started to spread to the entire

population, especially in the most underprivileged areas with a lack of land because tennis was only accessible to the wealthiest. By doing this, Frank Beal was able to persuade the city parks and recreation department, which chose to create many courts for this brand-new sport known as "Paddle Tennis" and disseminate it throughout the entire New York metropolitan region on his instructions [Almonacid-Cruz, 2011].

The first paddle tennis tournament was held in 1922, using the rules that the American Paddle Tennis Association would later approve. Since then, paddle tennis has been played in more than 500 cities across the nation and has become one of the most well-liked recreational activities, in addition to being a part of physical education programs in schools. It also serves as the foundation for developing skills and abilities in the early stages of learning tennis due to its simplicity [Almonacid-Cruz, 2011].

### **Platform Tennis**

In 1928, when paddle tennis was becoming more popular, Freseddenn Blanchardy and James Cogswell developed several improvements to their technique to practice it in the winter. The modifications were made to allow tennis players who lacked access to indoor facilities to practice during the winter. To this end, they built a wooden platform that would allow snow to accumulate on top of it easily and stay above ground level, allowing for outdoor play even after a snowfall [Sánchez-Alcaraz Martínez, 2013].

Later, a fence and an enclosure were added to the track to stop balls from leaving the court's boundaries. Blanchardy and Cogswell decided to play the game in pairs, permit the ball to bounce after it bounced on the fence, and substitute rubber balls and smaller paddles for the tennis ball. These decisions served as a model for the early stages of Padel tennis [Sánchez-Alcaraz Martínez, 2013].

Platform tennis had a very difficult time growing in the beginning, but evolution and changes were made, such as replacing the mesh wire with a metal structure and adding sand to the platform so that it could be played in the rain [Sánchez-Alcaraz Martínez, 2013].

As a result, in the beginning of the 1930s, this new sport modality became very well-liked in New York, New Jersey, Connecticut, and Washington, D.C., taking the place of tennis in the fall and winter [Hernández Vázquez, 1998].

### **The Existing Padel**

The most recent version of this sport, which is the most similar to the current one and which has recently been the version accepted by the FIP, claims that Padel tennis was invented in Acapulco (Mexico), in 1969, when Enrique Corcuera, using a wall on his property, installed some walls in the background and in the sides of a court with 20 meters long and 10 meters

wide so that the vegetation would not invade the land. The creation of this new sport, now known as Padel tennis, was made possible by these walls, which were 2 meters high on the sides and 3 meters high at the bottom, together with the installation of a net in the middle of the court, illustrated in Figure 1.2 [Castellote, 2012].

Later, because of the intense heat they experienced in Mexico, it was decided to lower the side wall's height and cover it with a wire mesh similar to what is today permitted. The game's scoring and ball rules were identical to those of tennis, with the exception that if the ball bounced off the walls after first hitting the ground, the play could continue. The rackets used were identical to those used in the United States for "Platform Tennis," which are shorter and have no ropes [Sánchez-Alcaraz Martínez, 2013].

Padel tennis was introduced to Argentina in 1975 by Argentine millionaire Julio Menditengui, a frequent visitor to Marbella. Within a few years, it had grown to be the second most popular sport there, with more than 4 million players and 10,000 courts, and had spread to other nearby nations like Brazil, Uruguay, Chile, and Paraguay [Almonacid-Cruz, 2011].



Figure 1.2: Recent padel [RoyalNorwich, 2022].

In the subsequent decades, Padel continued to escalate across Europe and Latin America, having countries like Portugal, Italy, France and Brazil welcoming the sport. In 2005, The World Padel Tour, an international professional Padel circuit, was founded and boosted the

sports visibility and captivated top players from around the world. The key factor for Padel's growth is its accessibility. Generally, since it is played in a smaller court, surrounded by walls and with the use of hard paddles, it is considered to be easier to snatch up for beginners when compared to tennis. Being considered simpler to play was a reason to an increase in the number of participants. In the past few years, Padel has grown into one of the fastest-growing sports in the world. As more people discover the excitement and fun of Padel, its global reach continues to expand, making it an ideal candidate for exploring the integration of artificial intelligence to enhance various aspects of the sport.

### 1.3.2 Rules and Gameplay

The Padel court is an intriguing fusion of the squash court's tight constraints and the tennis court's spacious openness. Its precise proportions of 20 meters in length and 10 meters in width encourage close player contact and quick exchanges. A net that rises 0.88 meters tall stands in the center of this rectangular court, neatly dividing it in half.

The service lines, which are placed at a distance of 6.95 meters on either side of the net, serve as the starting point of the game and invite strategic serves that establish the tone for each point. The Padel court is a compact but lively area that offers a middle ground between squash's snug setting and tennis' outdoor setting [IPF, 2017].

The court is encased and encircled by glass and metal mach walls. Wall areas surrounded by the glass material, as shown in Figure 1.3 in light blue, must be built using materials that allow for a consistent ball bounce and the remaining parts of the walls have to be built using materials that do not [IPF, 2017].

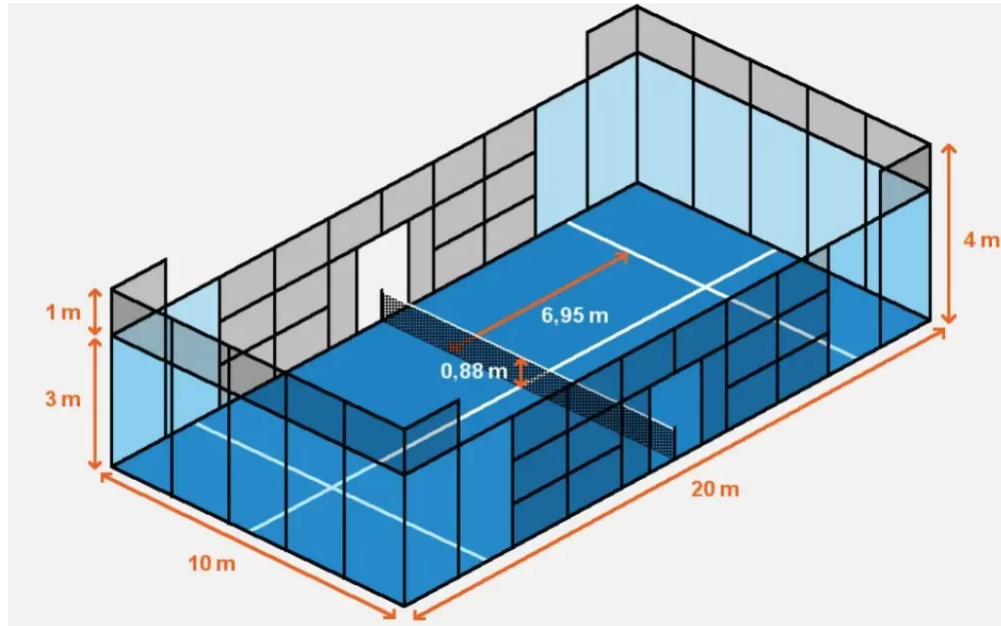


Figure 1.3: Padel court dimensions [PadelLab, 2023].

Due to the sport's heritage in racquet games, tennis and Padel's scoring systems are very similar to one another. In both Padel and tennis, a match is normally played to a certain number of games within a set, and to win a game, a player or team must hold a two-point lead. Similarly, a team will often win a set if they win six games, again with a two-game advantage. A tiebreaker is frequently used to decide who wins a set when the score is tied 6-6 [RookieRoad, 2023].

Every point begins with service. If the first service is invalid, the server may supply a second one. The player serving must begin the service by standing behind the service line, between the sidewall (service box) and the imagined extension of the central line of serve, and must remain there until the ball is served. To complete the serve, the server must bounce the ball on the ground inside the service box where they are standing [IPF, 2017]. The server's feet are not allowed to contact either the central line or the service line. At the time the ball is hit, the height of the ball being served must be at or below waist level, and the player must have at least one foot on the ground. The serving player should place the ball in the receiving box of service on the right side of the court, diagonally across the net. It must bounce inside the lines defining this box's boundaries.

When one side wins the point, it is time for the next service, and the ball must go into the receiver's box placed on his right side, and so on alternately. The first serve must go into the receiver's box located on his left side [IPF, 2017].

The service will be considered "net" if the ball served during a service touches the net or the supporting posts before landing in the receiver's service box and does not touch the



metallic fence before the second bounce. If the ball strikes a player or an object being carried or worn after touching the net or posts (if they are inside the game area), the service will also be considered "net."

Padel players can outmaneuver their opponents using a variety of motions and strategies in addition to the key serving component. According to [AllForPadel, 2023](#), these consist of:

- Forehand: Executed with the dominant side of the body, typically employing an open stance and aiming to control the direction and pace of the ball;
- Backhand: Executed with the non-dominant side of the body, involving a lateral swing that aims to return the ball to the opponent's side of the court;
- Volleys: Played close to the net and include striking the ball as it is in mid-air, just before it bounces on the court. Players utilize volleys to precisely place the ball and manage the game's tempo;
- Smashes: High-bouncing balls are taken advantage of with strong overhead shots, or smashes, that propel the ball down onto the opponent's court quickly and forcefully;
- Lobs: Lobs are high, arching shots designed to send the ball over the opponent's head, giving players the chance to alter the tempo of the game and tactically position their opponents;
- Drives: Drives are used to keep control of the net and apply pressure to the opposing team. Drives are flat, low shots aimed at the opponent's feet;
- Drop Shots: Drop shots are used to delicately deposit the ball just beyond the net with finesse and placement, frequently catching opponents off guard and motivating them to advance;
- Defensive Shots: When playing defense, athletes use moves like the "bandeja" and "vibora" to block difficult shots and keep their position on the court;
- Rally Play: Players must have rapid footwork and agile court coverage due to the fast-paced nature of Padel, moving both laterally and vertically in response to the flight of the ball.

The dynamic nature of Padel movements not only demands agility and finesse but also highlights the sport's blend of strategy and athleticism.

Each stroke is a careful balancing act between accuracy and planning. The right strokes can build up winning plays, yet even the smallest deviation from the ideal might lead to a mistake or fault which results in a lost point. A lost point happens when:

- During the service, the ball bounces outside of the service box of the receiver;
- During the service, the server, his partner, or any object they are wearing or carrying are struck by the ball;
- During the service, the ball makes two bounces in the receiver's service box before touching the court's metal fence;
- During the service, the ball bounces in the receiver's service box before bouncing out of the court via the court's gates without a safety zone and any legal out-of-court play;
- Before being returned, the ball bounces once more;
- If the athlete strikes the ball before it crosses the net;
- If a player returns the ball, either directly or by hitting it off the walls of their court, and it strikes the walls, the iron fence, or any other object not connected to or situated on the ground of the opponent's court without first bouncing;
- If a player returns a ball straight or off the walls of their court, and the ball touches the net or net posts before landing directly on one of the walls, the fence, or any other object that is not a part of or situated on the opponents court;
- If a participant strikes the ball twice (double hit);
- If a player strikes the ball and it makes contact with the metal fence, any area of the court's own side, or any object not intended for use on the court that is situated there [\[IPF, 2017\]](#).

In conclusion, the rules and gameplay of Padel create an exciting and dynamic sport that combines elements from tennis and squash while incorporating the unique use of walls. The emphasis on strategy, quick reflexes, and teamwork makes Padel an attractive sport for players of all ages and skill levels. With the growing interest in artificial intelligence applications in sports, the integration of AI in Padel presents promising opportunities to enhance player performance analysis, game strategy optimization, and overall player experience on the court.

## 1.4 AI

The cutting edge of technological development, AI, is reshaping how the interaction with technologies and the environment is done. At its heart, AI is a multidisciplinary science that aims to develop intelligent agents with human-like perception, reasoning, learning, and decision-making abilities. From its beginnings as a speculative quest into a transformational force with significant ramifications for many industries and facets of our daily lives, this discipline has experienced extraordinary growth.

The goal of AI is to create computers and algorithms that can carry out tasks that traditionally require human intelligence. These jobs cover a broad range, from language translation to picture identification, playing video games and making complicated decisions in industries like healthcare, finance, and autonomous systems. The adaptability of AI, which learns from data and experience to continuously improve performance, is one of its key characteristics. AI is a key and dynamic force in contemporary society because, at its core, it is about utilizing computer power and data to enhance human skills.

This section explores the types of AI, with a particular focus on its history and applications.

### 1.4.1 History and Evolution

The field of AI may be traced back to the middle of the 20th century when computer scientists and mathematicians started investigating the idea of building robots that might mimic human intelligence. The creation of the "Turing Test" in 1950 was one of the early significant achievements in AI [Turing, 1950].

This ground-breaking idea paved the way for determining if a machine can demonstrate intellect comparable to that of a human. The Dartmouth Workshop, which was put on by John McCarthy, Marvin Minsky, Nathaniel Rochester, and Claude Shannon in 1956, saw the beginning of AI as a discipline, with participants hoping to find solutions to issues like problem-solving and natural language understanding [McCarthy et al., 2006].

Significant symbolic AI advancements were made in the 1960s and 1970s when systems like the GPS and expert systems dominated [Newell et al., 1972]. There were "AI Winters" during this period, nevertheless, marked by unrealistic expectations and ensuing disappointments in AI's capabilities [Russell, 2016]. Despite obstacles, AI researchers persisted in seeking out fresh ideas.

The development of expert systems and neural networks in the middle of the 1980s rekindled interest in AI research. Geoffrey Hinton made a key advancement in machine learning by creating backpropagation for training artificial neural networks, which served as the basis for

contemporary deep learning [Rumelhart et al., 1986]. This innovation was crucial in getting over the constraints of early AI systems.

From the beginning of AI to the present, it has had a spectacular journey of technological development and interdisciplinary cooperation. AI research branched out into several related topics throughout the 1990s and the early 2000s, including computer vision, robotics, and natural language processing. In the latter half of the 20th century, probabilistic graphical models and Bayesian networks were developed, which made it possible for AI systems to perform more complex pattern recognition and reasoning [Pearl, 1988].

However, since the 2010s was when AI saw a significant rebirth, largely because of advances in deep learning and the accessibility of enormous datasets and computer capacity. Recurrent Neural Networks (RNN) for sequential data and Convolutional Neural Networks (CNN) for image recognition have both made significant advances in computer vision, speech recognition, and natural language processing [LeCun et al., 2015]. Applications of AI started to spread across many sectors, from banking to healthcare, altering how we approach issues and jobs.

AI is currently at the vanguard of technological advancement, with uses in everything from driverless automobiles to medical diagnostics. The fusion of AI with big data, cloud computing, and the Internet of Things (IoT) has driven the field to previously unheard-of heights, revolutionizing global economies and society. The development of AI throughout history is evidence of human creativity and our constant ambition to build intelligent devices that can complement our abilities and improve our understanding of the world.

#### 1.4.2 Types

AI is divided into several categories that include a variety of talents and goals. These categories are frequently referred to as specific or weak AI and general or strong AI. Weak AI, is created to thrive in a limited set of tasks or domains without having general intelligence or cognitive capabilities similar to those of humans. These extremely specialized AI systems are used often in many real-world contexts. Expert systems, which use predetermined rules and knowledge bases to make decisions and solve issues in particular domains like healthcare diagnosis or financial analysis, are examples of weak AI [Russell, 2010].

The goal of general AI, is to acquire an intellect level that is comparable to that of a human. This class of AI is capable of comprehending, learning, and adapting to a variety of tasks and domains while displaying cognitive skills that are similar to those of humans, such as reasoning, problem-solving, and self-awareness [Boström, 2014].

Additionally, supervised, unsupervised, and reinforcement learning are essential paradigms in machine learning, a branch of AI, as shown in Figure 1.4. An algorithm learns to translate

input data to predetermined output labels through supervised learning, which entails training AI models on labeled datasets. Regression analysis, natural language processing, and picture classification are just a few of the frequently used tasks.

While dealing with unlabeled data, unsupervised learning aims to identify patterns, correlations, or structures in the data. Unsupervised learning methods are often used for clustering and dimension reduction. In the third category, reinforcement learning, agents make decisions sequentially in an environment to maximize a cumulative reward. This paradigm is frequently applied in robotic autonomy and recommendation systems [Sutton and Barto, 2018].

### 1.4.3 Techniques and Algorithms

AI is a rapidly evolving field that includes a wide range of methods and algorithms intended to give machines the ability to carry out tasks that ordinarily require human intelligence. This subsection presents an overview of important AI methods and techniques, explaining how they function and outlining real-world uses for them.

- **Machine Learning:** A key component of AI is machine learning, which enables systems to learn from data and make predictions or judgments without having to be explicitly programmed. It consists of reinforcement learning, where agents learn by making mistakes, unsupervised learning, where patterns are found in unlabeled data, and supervised learning where models are trained on labeled data. Applications ranging from image identification to financial forecasting use machine learning methods like decision trees, support vector machines, and random forests [James et al., 2013].
- **Deep Learning:** Deep learning is a branch of machine learning that centers on deep neural networks, which are artificial neural networks with several layers. These networks are particularly good at tasks like audio and picture recognition, natural language processing, and autonomous driving because of their mastery of feature extraction and hierarchical learning. Among the structures frequently employed in deep learning are CNN and RNN [Goodfellow et al., 2016].
- **Neural Networks:** Neural networks are computerized representations of the brain. They are made up of interconnected nodes or neurons arranged in layers, with each layer processing and sending information to the one below it. Since they are adaptable, neural networks can be used in a variety of fields, such as recommendation systems, autonomous robotics, and speech and picture recognition [Haykin, 2009].
- **Natural Language Processing (NLP):** The interaction between computers and human language is the focus of the specialist field of NLP, which is part of AI. Machines can

comprehend, interpret, and produce human language thanks to NLP approaches. Chatbots for customer support, sentiment analysis in social media, machine translation, and speech recognition are examples of practical applications [Jurafsky and Martin, 2020].

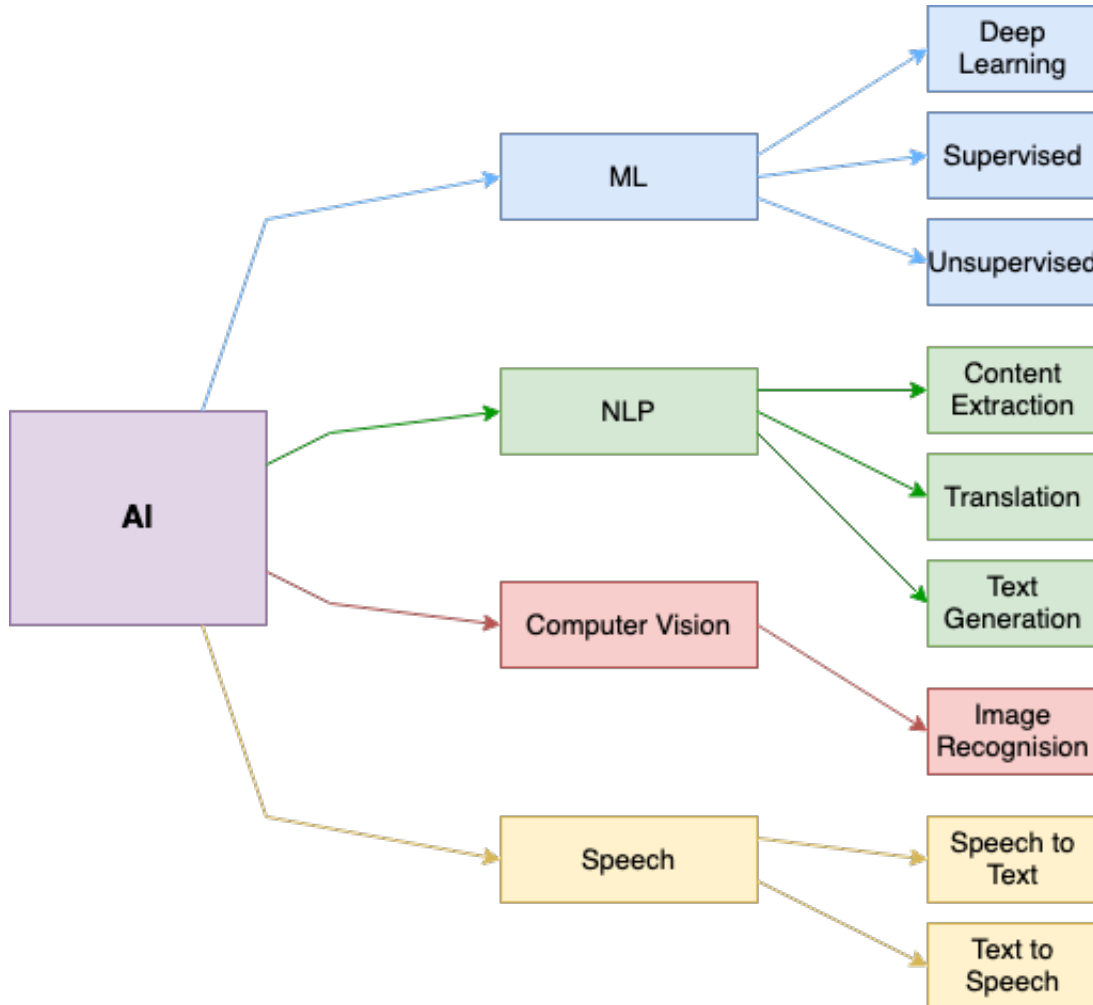


Figure 1.4: AI components.

When using machine learning techniques, massive datasets are used to train models to find patterns and relationships in the data. Machine learning algorithms, for instance, can examine patient data to forecast disease risk in medical diagnostics [Rajkomar et al., 2019]. To automatically extract complicated features from data, deep learning uses deep neural networks. Deep learning, for instance, assists autonomous vehicles in recognizing objects and making decisions about where to go in real time [Bojarski et al., 2016]. Neural networks can be used for image identification tasks, such as facial recognition in security systems or object detection in autonomous drones [LeCun et al., 2015]. They imitate the neural connections in the brain.

To process and comprehend text and speech, NLP approaches rely on algorithms like RNN and transformers. For the creation and understanding of natural language, they are used in virtual assistants like Siri and Google Assistant [Vaswani et al., 2017].

#### 1.4.4 Data and AI

Data is the lifeblood of AI, enabling applications to function powerfully and effectively. Data is the primary building block upon which AI systems are constructed, trained, and improved. The sheer amount, variety, and quality of data have a crucial impact on the effectiveness and dependability of AI systems. The proverb "garbage in, garbage out" succinctly illustrates the significance of data in AI. In essence, the caliber of the AI output is directly influenced by the caliber and relevance of the data employed.

The data-to-insight pipeline's first phase, data gathering, is of utmost importance. In order to complete this process, data must be gathered from a variety of sources, including sensors, social media sites, databases, and even user-generated material. The specific AI application and its goals will have a big impact on the data sources that are selected. In autonomous vehicles, data may originate from sensors, cameras, and GPS systems, but in healthcare AI, data may be gathered from patient records and medical devices [Goodfellow et al., 2016].

Data preprocessing, which entails preparing the obtained data for AI model training by cleaning, modifying, and arranging it, is also crucial. The performance of AI models can be negatively impacted by problems like missing values, outliers, and inconsistencies, which are mitigated through data pretreatment. To improve the quality and usability of the data, methods including feature engineering, data augmentation, and data standardization are frequently used in this phase [Goodfellow et al., 2016].

It is impossible to overestimate the quality of the dataset itself. Because an AI model can only be as good as the data it is trained on, employing incomplete or erroneous data can have unfavorable outcomes. Rigorous data validation, verification, and cleaning procedures are required to ensure data quality. Additionally, as biased datasets can reinforce and even exaggerate preexisting societal biases when used in AI applications, it is necessary to eliminate any potential biases in the data [Goodfellow et al., 2016].

In conclusion, it can not be overstated that data is the foundation of AI applications. To ensure the success and moral application of AI technologies, data collection, preparation, and maintenance of high-quality datasets are crucial. Since careful data management is necessary to fully utilize AI, it is an important subject for investigation and development in the field of artificial intelligence.

To review, the section developed by offering a fascinating look into the development and history of AI, demonstrating how it went from a theoretical idea to an ever-present force

influencing our digital age. AI was divided into its various subtypes, from specific AI to the aspirational broad AI. The power these tools offer for sensing, learning, reasoning, and decision-making were observed as the way through the complex maze of AI approaches and algorithms was made. Additionally, the crucial significance of data was emphasized, which serves as AI's lifeblood and is fundamentally linked to the techniques and technology that power AI's astounding powers.

## 1.5 Questions and Research Goals

The incorporation of AI presents a wide range of opportunities and problems in the world of Padel, a sport known for its dynamic nature and rising global appeal. The major objectives of this master's dissertation are to use AI to improve several parts of sport, from player performance enhancement to game analysis and audience engagement.

How AI technologies can be used to improve player performance in Padel is the main inquiry guiding this project. The analysis of Padel games should be automated. How can AI speed up the game analysis process, giving coaches and players tactical insights in real time? This research aims are centered on this subject as tools that provide precise and useful feedback are created, ultimately enhancing gameplay tactics.

An important area of investigation is the use of computer vision and data analytics in comprehending the nuances of Padel gameplay. How are players and the ball to be accurately and instantly tracked by computer vision technologies? How do data reveal previously hidden patterns and trends? These inquiries direct this study as creative answers that offer a thorough understanding of Padel pairings are created.



## 2 State of the art

With its ability to model and solve complicated problems, deep learning, a subfield of machine learning and a potent tool in AI, is transforming several industries. Deep learning trains models with good precision by using neural networks and complex algorithms, which are inspired by the structure and operation of the human brain. Deep learning, as opposed to traditional machine learning, does not require a lot of pre-processing to handle unstructured input, such as text and images. Deep learning algorithms continuously adapt and fit themselves to enhance accuracy by automating feature extraction and making use of methods like gradient descent and backpropagation. This versatility, along with its capacity to conduct supervised, unsupervised, and reinforcement learning, establishes deep learning as a cutting-edge technology with a wide range of uses in industries like speech recognition, image recognition, targeted advertising, and weather prediction.

### 2.1 CNN

CNN have emerged as a groundbreaking class of deep learning models revolutionizing various domains, including computer vision, natural language processing, and even audio analysis. CNN are specially designed to extract meaningful patterns and features from input data, making them particularly effective in image recognition, object detection, and image generation tasks. This section delves into the fundamentals of CNN, their architecture and training process.

#### Tensor

Knowing tensors is essential to understanding CNN because deep learning uses tensors as its primary data structure. A multi-dimensional array called a tensor may store complicated data, such as pictures or sequences.

To represent a vector, a symbol in boldface is used, e.g.,  $\mathbf{x} \in \mathbb{R}^d$  is a column vector with  $d$  elements. Capital letter is used to denote a matrix, e.g.,  $X \in \mathbb{R}^{h \times w}$  is a matrix with  $h$  rows and  $w$  columns. These concepts can be generalized to a higher-order matrices, i.e., tensor. For example,  $X \in \mathbb{R}^{h \times w \times d}$  is an order 3 tensor [Wu, 2017].

Tensors are employed in CNN to represent model parameters, intermediate feature maps, and input data. The fundamental function of CNN is the convolution operation, which works with tensors to enable the network to extract pertinent features from the input data.

#### Convolution Layer

The architecture of a CNN is specifically made to handle data that has a grid-like layout,

like images. It has a hierarchical structure made up of several layers that gradually pick up increasingly sophisticated features from the input data. Convolution layers, the fundamental components of a CNN, operate locally on the input data by applying tiny filters to extract significant characteristics. Then, pooling layers, which lower the spatial dimensions of the data and aid in capturing invariant features. Lastly, predictions based on the learned features are produced using fully connected layers.

A kernel is a small matrix used for the convolution operation, detecting specific patterns or features within input data. A kernel bank refers to a collection of multiple kernels in a single layer, each responsible for learning different features, enabling the network to extract complex representations from the input. The output of a convolution layer in a CNN is a set of feature maps, signaling certain features contained in the input image. The input image is convolved with one or more filters during the forward run of a CNN, leading to the production of numerous feature maps. Each one represents a different filter and contains the filter's response to the input image. Each feature map element represents the activation level of a given neuron in the network, with its value signifying the extent to which the relevant feature is detected in the input image [Murugan, 2017, Sakib et al., 2019].

For example, feature maps in the earliest layers of a CNN may capture low-level characteristics such as edges, lines, and corners. Deeper layers of the network may portray increasingly sophisticated elements such as forms, textures, or even full objects as the network evolves.

The number of feature maps in a convolution layer is a hyperparameter that can be changed during network building. Increasing the number of feature maps makes it easier to learn more detailed and abstract features, but it boosts computing costs and may lead to overfitting if the network becomes too large [Murugan, 2017].

Figure 2.1 illustrates an example of a  $3 \times 3$  filter (shown in green) applied to a  $5 \times 5$  input matrix, resulting in a  $3 \times 3$  feature map. The filter is placed over the top-left corner of the input matrix (highlighted in yellow) to compute the dot product for the first feature map element. The filter is then dragged to the right by one pixel to calculate the next element, and so on until all nine elements of the feature map are computed.

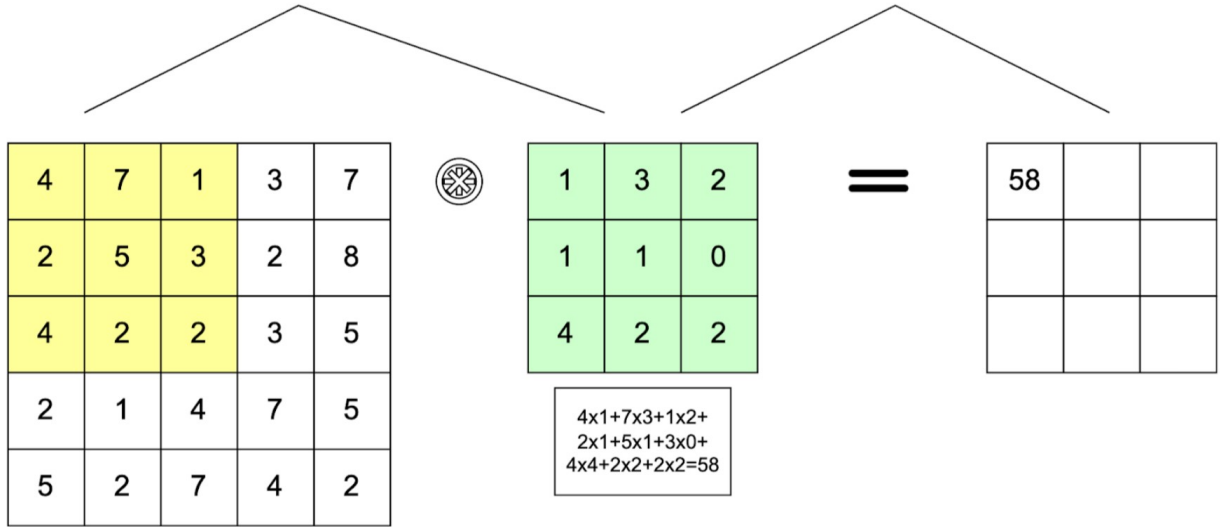


Figure 2.1: CNN Feature Map.

Convolution layers are made up of parallel feature maps that are produced by computing the element-wise dot product after moving different kernels, or feature detectors, across an input picture [Krizhevsky et al., 2012].

Smaller-sized kernel banks overlap on the input image during this sliding procedure, known as stride  $Z_s$ . The feature maps' dimensions are influenced by this overlap, which makes it easier for neighboring pixels in the image to share attributes like weight and bias. However, the efficacy of the learning method is limited because the usage of small kernels frequently results in inaccurate overlays. To overcome this constraint, the size of the input image is usually controlled using a Zero Padding  $Z_p$  procedure. By adding zeros to the input symmetrically, zero padding separately modifies the size of feature maps and kernels [Goodfellow et al., 2016].

Let  $H, W$  and  $C$  respectively correspond to the width, height and channel of an image and  $k_1, k_2$  and  $c$  respectively correspond to the width, height and depth of a convolution kernel.

A fixed-size input picture is slid over by a bank of kernel filters, or filter bank, with dimensions  $(k_1, k_2, c)$ , during algorithm training. Stride and zero padding are important parameters to regulate convolution layer dimensions. As a result, stacked feature maps are created to create the convolution layers.

Let  $H_1$  be the height of the convolution layer,  $W_1$  the width of convolution layer,  $D_1$  the depth of convolution layer and  $K_D$  the number of the kernel. The convolution layer's

dimension is:

$$Dim_c(H_1, W_1, D_1) = ((H + 2Z_p - k_1)/Z_s + 1), ((W + 2Z_p - k_2)/Z_s + 1), K_D.$$

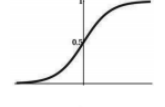
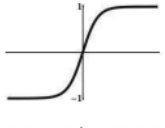
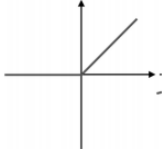
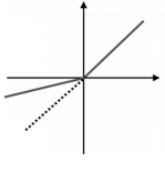

## Activation Functions

Through the processing of a collection of inputs, the activation function determines the output of a neuron. The activation function performs a non-linear change on the weighted sum of the linear net input values. A standard activation function produces an output of one or zero. It produces one, allowing the information to move on to the following layers, when the net input information exceeds the threshold value. The information is not delivered and the output is zero if the net input is less than the threshold. The activation function determines whether a neuron should fire by separating pertinent information from irrelevant data. Greater activation is correlated with a higher net input value [Murugan, 2017].

Many different kinds of activation functions have been created and utilized for various purposes. Some of the most popular ones are included in Table 2.1. The Rectified Linear Unit (ReLU), which replaces all negative values with zero and retains positive values, is the most often used activation function in a CNN. According to [Wu, 2017], enhancing the non linearity of a CNN is the aim of the ReLU function. Pixel values and semantic information have a very nonlinear relationship in computer vision tasks like item or scene recognition in images. It is also needed the CNN's transition from input to output to be very nonlinear to properly capture and depict this nonlinear mapping. The ReLU function is a vital component in bringing non linearity into the CNN architecture, despite its apparent simplicity. However, there are some issues with this activation function. The "dead ReLU" issue refers to a problem that can occur when using ReLU activation functions in neural networks, particularly deep neural networks. This problem arises when a ReLU neuron consistently outputs zero for all inputs during training and fails to update its weights during backpropagation. This can happen if the neuron's weights are initialized in such a way that it consistently produces negative outputs for all inputs. As a result, the gradient of the loss function with respect to the neuron's weights remains zero, preventing the weights from being updated during training. Consequently, the neuron remains "dead" and does not contribute to the learning process, leading to inefficient training and degraded performance of the neural network [Murugan, 2017].

The dead ReLU issue can negatively impact the convergence and effectiveness of the neural network, particularly in deep networks with many layers. To mitigate this problem, various strategies can be employed, such as using different activation functions like, for example the Leaky ReLU.

Table 2.1: Non-linear activation functions.

Name	Function	Derivative	Figure
Sigmoid	$\sigma(x) = \frac{1}{1+e^{-x}}$	$f'(x) = f(x)(1 - f(x))^2$	
tanh	$\sigma(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$f'(x) = 1 - f(x)^2$	
ReLU	$f(x) = \begin{cases} 0 & , x < 0 \\ x & , x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0 & , x < 0 \\ 1 & , x \geq 0 \end{cases}$	
Leaky ReLU	$f(x) = \begin{cases} 0.01x & , x < 0 \\ x & , x \geq 0 \end{cases}$	$f'(x) = \begin{cases} 0.01 & , x < 0 \\ 1 & , x \geq 0 \end{cases}$	
Softmax	$f(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}}$	$f'(x_i) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} - \frac{(e^{x_i})^2}{(\sum_{j=1}^n e^{x_j})^2}$	

## Pooling Layer

Down sampling is done by the pooling layer, which combines the output of neuron clusters from one layer into a single neuron in the following layer. To minimize data points and avoid overfitting, these actions take place following the non-linear activation. Furthermore, pooling eliminates undesired noise by acting as a smoothing procedure. The most popular operation is max pooling, which selects the maximum value of the correspondent cluster of neurons, while the other values are discarded [Murugan, 2017].

Let  $H_2$  be the height of the pooling layer,  $W_2$  the width of pooling layer,  $D_2$  the depth of pooling layer and  $k$  the width and height of the pooling layer kernel. The dimension of the pooling layer can be computed as follows when pooling layers are created with  $D_N$  kernel windows.

$$Dim_c(H_2, W_2, D_2) = ((H_1 - k)/Z_s + 1), ((W_1 - k)/Z_s + 1), D_N$$

## Fully Connected Layer

The pixels in the pooling layers that come after are stretched into a single-column vector. For classification, these vectorized and concatenated data points are subsequently fed into

dense layers, called fully connected layers [Ciresan et al., 2011]. In the domain of picture classification challenges, this limited design appears to be highly effective, beating standard machine learning methods [Schmidhuber, 2015].

## Loss Function

A loss function gives an event involving one or more variables a real value that represents a cost. It measures the difference between the expected value  $\hat{y}_i^{L+1}$  and the actual value  $y_i$ , acting as a statistic to assess the model's performance. As the loss function value drops, the model becomes more effective [Murugan, 2017]. According to [Murugan, 2017], several loss functions have been developed to suit to various purposes.

**Mean Squared Error** The Mean Squared Error (MSE), commonly known as the Quadratic Loss Function, is largely used in linear regression models to evaluate performance. The MSE is derived using the computed output value  $\hat{y}_i^{L+1}$  for the  $i$ -th training sample and the labeled value  $y_i$  and it is given by,

$$L(\hat{y}^{L+1}, y) = \frac{1}{t} \sum_{i=1}^t (y_i - \hat{y}_i^{L+1})^2$$

with  $t$  equal to the total number of training samples.

One disadvantage of MSE is its tendency to demonstrate slow learning speed (slow convergence) when combined with the Sigmoid activation function.

## Mean Squared Logarithmic Function

$$L(\hat{y}^{L+1}, y) = \frac{1}{t} \sum_{i=1}^t (\log(y_i + 1) - \log(\hat{y}_i^{L+1} - 1))^2$$

**$L_2$  Loss Function** The  $L_2$  loss function is the square root of the  $L_2$  norm, which represents the difference between the real labeled value and the computed value based on the net input. This is expressed as follows,

$$L(\hat{y}^{L+1}, y) = \sum_{i=1}^t (y_i - \hat{y}_i^{L+1})^2$$

**$L_1$  Loss Function** The  $L_1$  loss function denotes the sum of absolute errors, which are the absolute discrepancies between the real labeled value and the computed value generated

from the net input. The mathematical expression is as follows,

$$L(\hat{y}^{L+1}, y) = \sum_{i=1}^t |y_i - \hat{y}_i^{L+1}|$$

**Mean Absolute Error** Mean Absolute Error measures the similarity between predicted and actual values, as represented by,

$$L(\hat{y}^{L+1}, y) = \frac{1}{t} \sum_{i=1}^t |y_i - \hat{y}_i^{L+1}|$$

**Mean Absolute Percentage Error**

$$L(\hat{y}^{L+1}, y) = \frac{1}{t} \sum_{i=1}^t \left| \left( \frac{y_i - \hat{y}_i^{L+1}}{y_i} \right) \right| \times 100$$

**Cross Entropy** The Cross Entropy loss function is the most commonly used, and it is given below,

$$L(\hat{y}^{L+1}, y) = -\frac{1}{t} \sum_{i=1}^t ((y_i) \log(\hat{y}_i^{L+1}) + (1 - y_i) \log(1 - \hat{y}_i^{L+1})). \quad (2.1)$$

**Forward Propagation**

Forward propagation is the process through which input data is passed through a neural network, layer by layer, to compute its output. In forward propagation, the input values are multiplied by randomly initialized weights, and then each neuron's connection's randomly initialized bias values are added. The total of the products from all of the neurons comes next. The net input value is then transformed using non-linear activation functions [Murugan, 2017](#).

Let  $m$  and  $n$  indices represent spatial coordinates in a color channel  $c$  and let  $u$  and  $v$  be the pixels of a kernel. Let  $p$  and  $q$  represent the number of the convolution kernel and the number of convolution layer, respectively. Over the image  $I_{m,n}$ , a kernel bank, represented as  $k_{u,v}^{p,q}$ , is slid with a stride value of 1 and a zero padding value of 0. This is called the convolution operation and can be represented as follows:

$$(I \otimes K)_{ij} = \sum_{i=1}^m \sum_{j=1}^n K_{m,n} I_{i+m,j+n}.$$

A non-linear activation function, represented by the symbol  $\sigma$ , is then applied to these feature maps adding a bias value  $b$  to the convoluted part. The following equation can be used to compute the feature maps of the convolution layer  $C_{m,n}^{p,q}$ :

$$C_{m,n}^{p,q} = \sigma \left( \sum_{i=1}^m \sum_{j=1}^n I_{(i-u, j-v)} \cdot k_{u,v}^{p,q} + b^{p,q} \right).$$

The pooling layer  $P_{m,n}^{p,q}$ , is created by taking, for example, the maximum-valued pixels ( $m$  and  $n$ ) out of the convolution layers. The following expression can be used to express the pooling layer calculation:

$$P^{p,q} = \max (C_{m,n}^{p,q}). \quad (2.2)$$

Then the pooling layer  $P^{p,q}$  results are concatenated in an extended vector with length  $p \times q$ . For classification, this vector is subsequently fed into fully connected dense layers. Afterward, the  $a_i^{l-1}$  layer's vectorized data points are represented by:

$$a^{l-1} = f (P^{p,q}).$$

The extended vector is input into a sequence of fully connected layers and stretch from the  $l^{th}$  layer to the  $(L + 1)^{th}$  layer. These are built with  $L$  layers and  $n$  neurons each, where  $l$  is the first layer,  $L$  is the last layer, and  $(L + 1)$  is the classification layer, as illustrated in Figure 2.2 [Murugan, 2017].

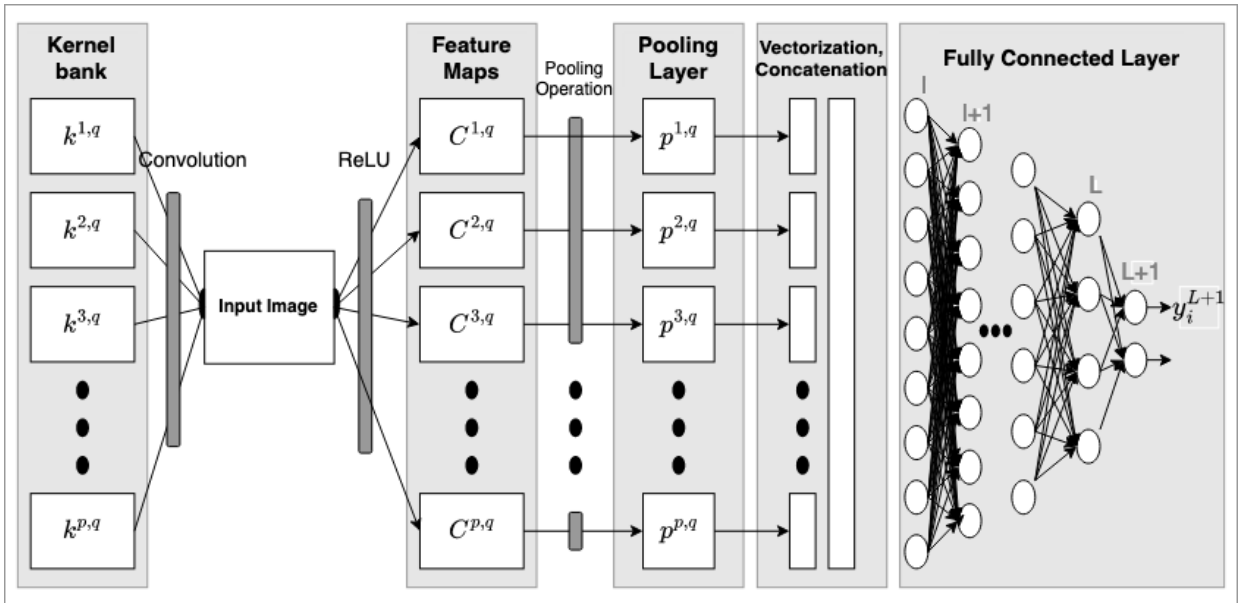


Figure 2.2: CNN Architecture.



In the forward run, the layer progression is stated as follows:

$$\begin{aligned} z_1^l &= w_{11}^l a_1^{l-1} + w_{12}^l a_2^{l-1} + \dots + w_{1j}^l a_j^{l-1} + \dots + b_j^l \\ z_2^l &= w_{21}^l a_1^{l-1} + w_{22}^l a_2^{l-1} + \dots + w_{2j}^l a_j^{l-1} + \dots + b_j^l \\ z_i^l &= w_{i1}^l a_1^{l-1} + w_{i2}^l a_2^{l-1} + \dots + w_{ij}^l a_j^{l-1} + \dots + b_j^l \end{aligned}$$

$$\begin{bmatrix} z_1^l \\ \vdots \\ z_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} w_{11}^l & w_{12}^l & w_{13}^l & \dots & w_{1n}^l \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{i1}^l & w_{i2}^l & w_{i3}^l & \dots & w_{in}^l \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix} \begin{bmatrix} a_1^{l-1} \\ \vdots \\ a_i^{l-1} \\ \vdots \end{bmatrix} + \begin{bmatrix} b_1^l \\ \vdots \\ b_i^l \\ \vdots \end{bmatrix}.$$

For example, consider a single neuron ( $j$ ) that is located in a fully connected layer at layer  $l$ . Weights  $w_{i,j}$  and bias values  $b_j^l$ , respectively, are used to multiply and add the input values, which are denoted as  $a_i^{l-1}$ .

$$\begin{aligned} z_j^l &= w_{1j}^l a_1^{l-1} + w_{2j}^l a_2^{l-1} + \dots + w_{ij}^l a_i^{l-1} + \dots + b_j^l \\ &= \sum_{i=1}^n w_{ij}^l a_i^{l-1} + b_j^l. \end{aligned}$$

Thereafter, a non-linear activation function  $\sigma$  is applied to the net input value  $z_i^l$ . This procedure then determines the  $l^{\text{th}}$  output layer value for the neuron  $j$ ,  $a_j^l$ :

$$a_j^l = \sigma \left( \sum_{i=1}^n w_{ij}^l a_i^{l-1} + b_j^l \right).$$

Thus, the  $l^{\text{th}}$  layer has output equal to:

$$\begin{aligned} a^l &= \sigma \left( (W^l)^T a^{l-1} + b^l \right) \\ &= \sigma (z^l) \end{aligned}$$

$$a^l = \begin{bmatrix} a_1^l \\ \vdots \\ a_i^l \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma(z_1^l) \\ \vdots \\ \sigma(z_i^l) \\ \vdots \end{bmatrix}$$

$$W^l = \begin{bmatrix} w_{1j}^l \\ \vdots \\ w_{ij}^l \\ \vdots \end{bmatrix}.$$

Likewise, the output value of the final layer  $L$  is determined by the following expression:

$$\begin{aligned} a^L &= \sigma \left( (W^L)^T a^{L-1} + b^L \right) \\ &= \sigma (z^L) \end{aligned}$$

$$a^L = \begin{bmatrix} a_1^L \\ \vdots \\ a_i^L \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma (z_1^L) \\ \vdots \\ \sigma (z_i^L) \\ \vdots \end{bmatrix}.$$

Extending this concept to classification layers, the ultimate predicted output value  $\hat{y}_i^{L+1}$  for the neuron ( $i$ ) at the  $L + 1$  layer can be articulated as:

$$\hat{y}_i^{L+1} = \sigma (W^L \dots \dots \sigma (W^2 \sigma (W^1 a^1 + b^1) + b^2) \dots \dots + b^L).$$

Let  $t$  be the total number of training samples. According to [Murugan, 2017](#), having the predicted value denoted as  $\hat{y}_i^{L+1}$  and the actual labeled value  $y_i$ , the model's performance can be evaluated using the cross-entropy loss function, as expressed in Equation [2.1](#):

$$L (\hat{y}_i^{L+1}, y_i) = -\frac{1}{t} \sum_{i=1}^t (y_i \log (\hat{y}_i^{L+1}) + (1 - y_i) \log (1 - \hat{y}_i^{L+1})).$$

## Backward Run And Parameter Updates

During the backward run in the training of a neural network, both the weights and biases are updated based on the computed gradients of the loss function with respect to these parameters. This process involves propagating the error gradients of the loss function backward through the network, from the output layer to the input layer, and using these gradients to adjust the weights and biases in each layer. Importantly, reusing partial derivatives between layers via the chain rule improves the computational efficiency of gradient calculation at each layer. This all adds to the goal of minimizing the loss function [Rumelhart et al., 1986](#), [Pineda, 1987](#), [LeCun et al., 1998](#). Parameters such as  $W^{L+1}, b^{L+1}, W^l, b^l, k^{p,q}$  and  $b^{p,q}$  must

be updated during this procedure in order to iteratively reduce the cost function.

The derivative of the loss function for the  $i^{th}$  neuron at the  $(L + 1)^{th}$  classification layer concerning the predicted values  $\hat{y}_i^{L+1}$  is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} &= -\frac{1}{t} \sum_{i=1}^t \frac{\partial (y_i \log(\hat{y}_i^{L+1}) + (1 - y_i) \log(1 - \hat{y}_i^{L+1}))}{\partial \hat{y}_i^{L+1}} \\ &= \frac{1}{t} \sum_{i=1}^t \frac{\partial (-1 (y_i \log(\hat{y}_i^{L+1}) + (1 - y_i) \log(1 - \hat{y}_i^{L+1})))}{\partial \hat{y}_i^{L+1}} \\ &= \frac{1}{t} \sum_{i=1}^t \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}}. \end{aligned}$$

The derivative of the loss function for all neuron at the  $(L + 1)^{th}$  classification layer concerning the predicted values  $\hat{y}_i^{L+1}$  is expressed as:

$$\begin{bmatrix} \frac{\partial L(\hat{y}_1^{L+1}, y_1)}{\partial \hat{y}_1^{L+1}} \\ \frac{\partial L(\hat{y}_2^{L+1}, y_2)}{\partial \hat{y}_2^{L+1}} \\ \vdots \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{1}{t} \sum_{i=1}^t \frac{-y_1}{\hat{y}_1^{L+1}} + \frac{1 - y_1}{1 - \hat{y}_1^{L+1}} \\ \frac{1}{t} \sum_{i=1}^t \frac{-y_2}{\hat{y}_2^{L+1}} + \frac{1 - y_2}{1 - \hat{y}_2^{L+1}} \\ \vdots \\ \frac{1}{t} \sum_{i=1}^t \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}} \\ \vdots \end{bmatrix}.$$

The derivative of the loss function for the  $i^{th}$  neuron at the  $L^{th}$  classification layer concerning the weight  $w_i^L$  is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \frac{\partial \hat{y}_i^{L+1}}{\partial w_i^L} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}} \right) \left( \frac{\partial \hat{y}_i^{L+1}}{\partial w_i^L} \right) \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}} \right) \left( \frac{\partial a_i^{L+1}}{\partial w_i^L} \right) \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}} \right) \left( \frac{\partial \sigma(z_i^{L+1})}{\partial w_i^L} \right) \end{aligned}$$

$$\begin{aligned}
\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{t=1}^i \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) \sigma'(z_i^{L+1}) \\
\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) \sigma' \left( \sum_1^i w_i a^{L-1} + b^L \right). \tag{2.3}
\end{aligned}$$

In the final layer  $L$ , a sigmoid activation function is employed for nonlinear transformation. As indicated in Table [2.1](#), the sigmoid activation function is denoted as:

$$\begin{aligned}
\sigma(z_i^{L+1}) &= \frac{1}{1 + e^{-z_i^{L+1}}} \\
\frac{\partial \sigma(z_i^{L+1})}{\partial z_i^{L+1}} &= \sigma(z_i^{L+1}) (1 - \sigma(z_i^{L+1})). \tag{2.4}
\end{aligned}$$

By substituting the Equation [\(2.4\)](#) in Equation [\(2.3\)](#) the following is obtained:

$$\begin{aligned}
\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) \left( \sigma \left( \sum_1^i w_i a^{L-1} + b^L \right) \right) \\
&\qquad \qquad \qquad \left( 1 - \sigma \left( \sum_1^i w_i a^{L-1} + b^L \right) \right) \\
\hat{y}_i^{L+1} &= a_i^{L+1} = \sigma(z_i^{L+1}) \tag{2.5}
\end{aligned}$$

$$\begin{aligned}
\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} &= \frac{1}{t} \sum_{t=1}^i \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\sigma \left( \sum_1^i w_i a^{L-1} + b^L \right)} \right) \left( \sigma \left( \sum_1^i w_i a^{L-1} + b^L \right) \right) \\
&\qquad \qquad \qquad \left( 1 - \sigma \left( \sum_1^i w_i a^{L-1} + b^L \right) \right) = \frac{1}{t} \sum_{t=1}^i \hat{y}_i^{L+1} \left( \sigma \left( \sum_1^i w_i a^{L-1} + b^L - y_i \right) \right).
\end{aligned}$$

Thus, the partial derivative of the loss function with respect to the weights of every neuron

in the  $L^{th}$  layer is formulated as:

$$\frac{\partial L(\hat{y}^{L+1}, y)}{\partial w_i^L} = \begin{bmatrix} \frac{\partial L(\hat{y}_1^{L+1}, y_1)}{\partial w_1^L} \\ \frac{\partial L(\hat{y}_2^{L+1}, y_2)}{\partial w_2^L} \\ \vdots \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^L} \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{1}{t} \sum_{i=1}^t \hat{y}_1^{L+1} (\sigma(z_1^{L+1} - y_i)) \\ \frac{1}{t} \sum_{i=1}^t \hat{y}_2^{L+1} (\sigma(z_2^{L+1} - y_i)) \\ \vdots \\ \frac{1}{t} \sum_{i=1}^t \hat{y}_i^{L+1} (\sigma(z_i^{L+1} - y_i)) \\ \vdots \end{bmatrix}.$$

The partial derivative of the loss function for the  $i^{th}$  neuron at the  $L^{th}$  classification layer concerning the bias  $b_i^L$  is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b_i^L} &= \frac{1}{t} \sum_{i=1}^t \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \frac{\partial \hat{y}_i^{L+1}}{\partial b_i^L} \\ &= \sigma(z_i^{L+1}) - y_i. \end{aligned}$$

Therefore, the partial derivative of the cost function with respect to the bias of every neuron at layer  $L$  is expressed as:

$$b^L = \begin{bmatrix} \frac{\partial L(\hat{y}_1^{L+1}, y_1)}{\partial b_1^L} \\ \frac{\partial L(\hat{y}_2^{L+1}, y_2)}{\partial b_2^L} \\ \vdots \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b_i^L} \\ \vdots \end{bmatrix} = \begin{bmatrix} \sigma(z_1^{L+1}) - y_1 \\ \sigma(z_2^{L+1}) - y_2 \\ \vdots \\ \sigma(z_i^{L+1}) - y_i \\ \vdots \end{bmatrix}.$$

Similarly, the partial derivatives of the loss function with respect to all hidden neurons and hidden layers can be computed. ReLU is employed across all hidden layers from  $l - 1$  to  $L_1$ . The partial derivative of the loss function with respect to the weight of the  $i^{th}$  neuron in the first layer  $l$  of the fully connected dense layer is:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^l} &= \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \frac{\partial \hat{y}_i^{L+1}}{\partial w_i^l} \\ &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1 - y_i}{1 - \hat{y}_i^{L+1}} \right) \sigma' \left( \sum_{i=1}^t w_i a^{l-1} + b^l \right). \end{aligned}$$

Since the derivative of the ReLU function is defined as:

$$\sigma'(x) = \begin{cases} 0 & , x < 0 \\ 1 & , x \geq 0 \end{cases}$$

and in the case when  $z > 0$ :

$$\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^l} = \frac{y_i - z_i^l}{z_i^l(1 - z_i^l)}.$$

Thus, the partial derivative of the loss function with respect to the weights of every neuron in the  $l$  layer is formulated as:

$$W^l = \begin{bmatrix} \frac{\partial L(\hat{y}_1^{L+1}, y_1)}{\partial w_1^l} \\ \frac{\partial L(\hat{y}_2^{L+1}, y_2)}{\partial w_2^l} \\ \vdots \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial w_i^l} \\ \vdots \end{bmatrix} = \begin{bmatrix} \frac{y_1 - z_1^l}{z_1^l(1 - z_1^l)} \\ \frac{y_2 - z_2^l}{z_2^l(1 - z_2^l)} \\ \vdots \\ \frac{y_i - z_i^l}{z_i^l(1 - z_i^l)} \\ \vdots \end{bmatrix}.$$

The partial derivative of the loss function for the  $i^{th}$  neuron at the  $l$  classification layer concerning the bias  $b_i^l$  is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b_i^l} &= \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \frac{\partial \hat{y}_i^{L+1}}{\partial b_i^l} \\ &= \sigma(z^{l-1}) - y_i \end{aligned}$$

where  $\sigma$  is the ReLU function, thus, if  $z_i > 0$  then:

$$\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b_i^l} = z_i^{l-1} - y_i.$$

Therefore, the partial derivative of the cost function with respect to the bias of every neuron

at layer  $l$  is expressed as:

$$b^l = \begin{bmatrix} \frac{\partial L(\hat{y}_1^{L+1}, y_1)}{\partial b_1^l} \\ \frac{\partial L(\hat{y}_2^{L+1}, y_2)}{\partial b_2^l} \\ \vdots \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b_i^l} \\ \vdots \end{bmatrix} = \begin{bmatrix} z_1^{l-1} - y_1 \\ z_2^{l-1} - y_2 \\ \vdots \\ z_i^{l-1} - y_i \\ \vdots \end{bmatrix}.$$

To conclude the learning process of CNN, updating the weights of the kernel bank and bias values is essential not only in convolution layers but also in pooling layers. The partial derivative of the loss function with respect to the input value  $a_i^{l-1}$  is:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial a_i^{l-1}} &= \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial \hat{y}_i^{L+1}} \frac{\partial \hat{y}_i^{L+1}}{\partial a_i^{l-1}} \\ &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) \frac{\partial \hat{y}_i^{L+1}}{\partial a_i^{l-1}} \\ &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) \frac{\partial (w_i^L a^L + b^L)}{\partial a_i^{l-1}} \\ &= \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) w_i^l. \end{aligned}$$

At  $(l-1)^{th}$  layer for all input values  $a^{l-1}$ ,

$$\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial a_i^{l-1}} = \frac{1}{t} \sum_{i=1}^t \left( \frac{-y_i}{\hat{y}_i^{L+1}} + \frac{1-y_i}{1-\hat{y}_i^{L+1}} \right) W^L.$$

Reconfiguring the long vector  $\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial a_i^{l-1}}$ ,

$$P^{p,q} = \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial a_i^{l-1}}.$$

The main purpose of a pooling layer is to decrease the number of parameters and mitigate overfitting in the model. Consequently, pooling layers do not contribute to the learning process. The error in the pooling layer is determined by selecting the winning unit, which comprises a single value. Due to the absence of parameters that require updating in the

pooling layer, upsampling can be performed to retrieve

$$\frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} = P^{p,q}.$$

The partial derivative of the loss function concerning the convolution kernel  $k_{u,v}^{p,q}$  is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial k_{u,v}^{p,q}} &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} \frac{\partial C_{m,n}^{p,q}}{\partial k_{u,v}^{p,q}} \\ \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial k_{u,v}^{p,q}} &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} \frac{\partial (\sum_1^u \sum_1^v I_{i-u, j-v} k_{u,v}^{p,q} + b^{p,q})}{\partial k_{u,v}^{p,q}} \\ &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} I_{m-u, j-v}. \end{aligned}$$

The partial derivative of the loss function concerning the bias  $b^{p,q}$  of the convolution kernel is expressed as:

$$\begin{aligned} \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial b^{p,q}} &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} \frac{\partial C_{m,n}^{p,q}}{\partial b^{p,q}} \\ &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}} \frac{\partial (\sum_1^u \sum_1^v I_{i-u, j-v} k_{u,v}^{p,q} + b^{p,q})}{\partial b^{p,q}} \\ &= \sum_{i=1}^m \sum_{j=1}^n \frac{\partial L(\hat{y}_i^{L+1}, y_i)}{\partial C_{m,n}^{p,q}}. \end{aligned}$$

To minimize the loss function, the learning parameters must be updated at each iteration using gradient descent. The weight and bias update for the fully connected layer  $L + 1$  is represented as follows:

$$\begin{aligned} W^{L+1} &= W^{L+1} - \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial W^L} \\ b^{L+1} &= b^{L+1} - \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial b^L}. \end{aligned}$$



The weight and bias update for the fully connected layer  $l$  is represented as follows:

$$W^l = W^l - \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial W^l}$$

$$b^l = b^l - \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial b^l}.$$

The weight and bias update for the convolution kernel  $l$  is represented as follows:

$$k^{p,q} = k^{p,q} - \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial k_{u,v}^{p,q}}$$

$$b^{p,q} = \alpha \frac{\partial L(\hat{y}^{L+1}, y)}{\partial b^{p,q}}.$$

By iteratively performing forward and backward passes through the network and adjusting the weights and biases accordingly, the network learns to better approximate the desired output for a given input. This iterative process continues until the network converges to a set of weights and biases that minimize the loss function and result in satisfactory performance on the training data.

## 2.2 YOLO Object Detection Model

Object detection is a computer vision task that involves finding and identifying things of interest in pictures or videos. Technology for object detection has advanced significantly. The You Only Look Once (YOLO) algorithms have gained popularity in computer vision. Their appeal lies in their high accuracy and small model size. YOLO models can be trained on a single GPU, making them accessible to a wide range of developers. From the groundbreaking real-time detection capabilities of YOLOv1 to YOLOv8, each iteration has improved upon the accuracy and addressed the shortcomings of its forerunners. With its cutting-edge methods and performance enhancements, YOLOv4 stood out as a crucial turning point in the YOLO series. The latest version, YOLOv8, developed by Ultralytics, offers advanced features for object identification, image categorization, and segmentation. This cutting-edge object identification model is capable of real-time performance and great accuracy. Due to its high efficiency, YOLO is a strong option for applications where real-time speed is important. YOLO remains at the vanguard of object detection as it develops, pushing the limits of accurate and real-time object recognition. This chapter delves into the fundamentals of YOLO versions and their architecture.

## History and Evolution

The YOLO model revolutionized object detection by enabling real-time detection. It was created to solve the shortcomings of current object detection techniques, such as their long processing times and poor accuracy. By utilizing a single neural network to simultaneously forecast bounding boxes and class probabilities, YOLO sought to enable real-time object detection. This method generated impressive speed increases while eliminating the requirement for region proposal models [Redmon et al., 2016]. It emphasizes the necessity of quick and precise object detection for uses in robotics, surveillance, and autonomous driving.

YOLO is able to complete the detection task with a single network pass, in contrast to earlier methods that either used sliding windows followed by a classifier that had to be run hundreds or thousands of times per image, or the more sophisticated approaches that split the task into two steps, where the first step detects potential regions with object or region proposals and the second step runs a classifier on the proposals [Terven and Cordova-Esparza, 2023].

YOLO has gone through multiple versions since its beginning, as shown in Figure 2.3, each with its developments and enhancements. By examining the development and variations across many YOLO iterations, this analysis hopes to give information on the advancements made in object-detecting technologies [Terven and Cordova-Esparza, 2023].

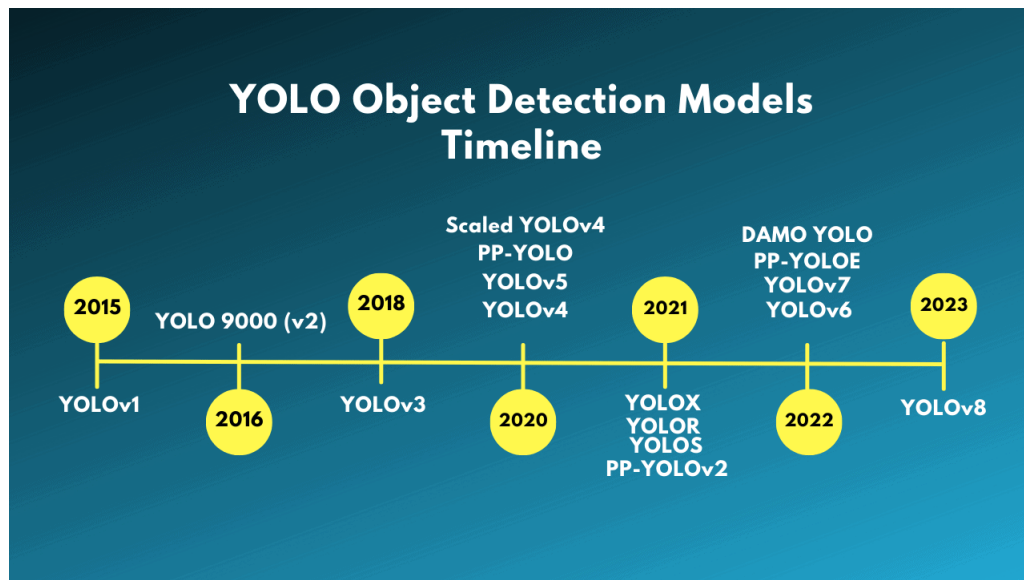


Figure 2.3: YOLO object detection models timeline [LearnOpenCV, 2023].

- YOLOv1: According to [Hussain, 2023], the algorithm's first iteration, which was released in 2015, was known as YOLOv1. To directly predict bounding boxes and class probabilities, it used a single CNN. Although YOLOv1 was renowned for its excellent real-time detection speed, its coarse feature maps made it ineffective at detecting small

objects. It did, however, set the stage for further iterations to overcome these shortcomings.

YOLOv1 unified the object detection procedures by detecting all the bounding boxes concurrently. To accomplish this, YOLO divides the input image into a  $S \times S$  grid and predicts  $B$  bounding boxes of the same class, along with its confidence for  $C$  different classes per grid element. Each bounding box prediction consists of five values:  $P_C, b_x, b_y, b_h, b_w$  where  $P_C$  is the confidence score for the box that shows how confident the model is that the box contains an object and how accurate the box is. The  $b_x$  and  $b_y$  coordinates are the centroid of the box concerning the grid cell, while  $b_h$  and  $b_w$  are the height and width of the box related to the complete image. YOLO returns a tensor of  $S \times S \times (B \times 5 + C)$  [Terven and Cordova-Esparza, 2023].

Figure 2.4 depicts a simplified output vector for eight values using a three-by-three grid,  $S = 3$ , three classes,  $C = 3$ , and one bounding box,  $B = 1$ . In this simplified scenario, YOLO's output would be  $3 \times 3 \times 8$ .

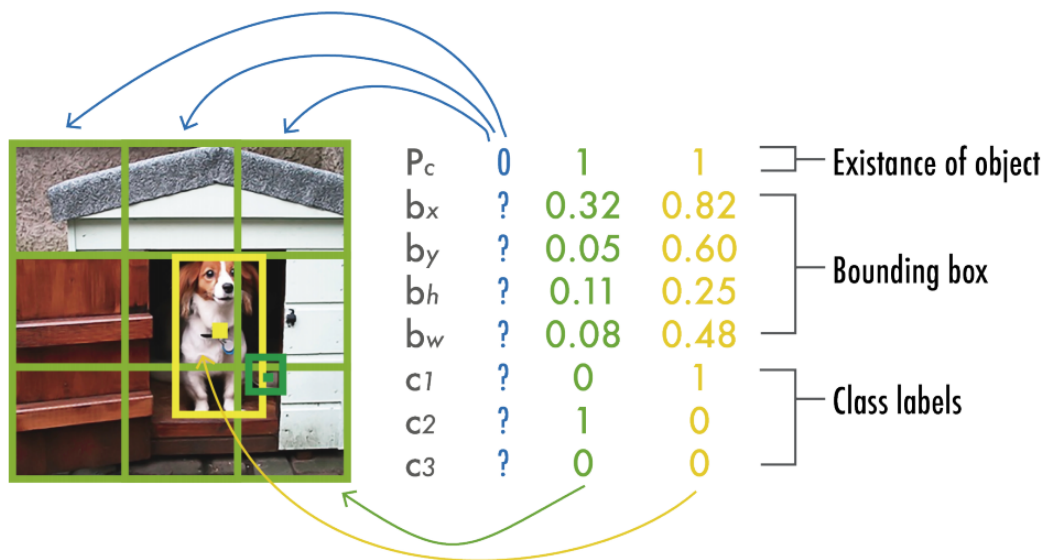


Figure 2.4: YOLO prediction example [Terven and Cordova-Esparza, 2023].

The YOLOv1 architecture is made up of 24 convolution layers, which are followed by two fully connected layers that predict the bounding box coordinates and probabilities. Except for the final layer, which utilized a linear activation function, all layers used Leaky ReLU [Maas et al., 2013].

- YOLOv2: Introduced in 2016 by [Redmon and Farhadi, 2017], YOLOv2 addressed the problem of small item identification by utilizing anchor boxes. This version maintained

real-time detection speed while achieving cutting-edge performance. Additionally, the YOLOv2 offered multi-scale training, enabling the accurate recognition of objects at various scales.

According to [Terven and Cordova-Esparza, 2023], some of the improvements over the original YOLO were the following:

1. Batch normalization enhanced convergence and functions as a regularizer to reduce overfitting.
2. Classifier with high resolution. Pre-trained the model with ImageNet (a large visual database designed for use in visual object recognition software research) at  $224 \times 224$ , much like YOLOv1. They finetuned the model for ten epochs on ImageNet with a resolution of  $448 \times 448$ , boosting network performance on higher resolution input.
3. Fully convolution Removed the dense layers in favor of a completely convolution architecture.
4. Training on multiple scales. Because YOLOv2 does not employ fully connected layers, the inputs can be of varying sizes. To make YOLOv2 resistant to multiple input sizes, the authors trained the model at random, increasing the input size from  $320 \times 320$  to  $608 \times 608$  every 10 batches.

With all of these enhancements, YOLOv2 obtained an average precision (AP) of 78.6% on the PASCAL VOC2007 dataset [Everingham et al., ], compared to 63.4% for YOLOv1.

YOLOv2's backbone architecture consists of 19 convolution layers and five max-pooling layers. The last four convolution layers are replaced by a single convolution layer with 1000 filters, followed by a global average pooling layer in the object classification head [Terven and Cordova-Esparza, 2023].

- YOLOv3: With the introduction of various architectural enhancements, YOLOv3, released in 2018, greatly improved object detection accuracy. The model can now detect objects of varied sizes thanks to the three alternative detection scales introduced in YOLOv3.

It used strided convolutions to replace all max-pooling layers and added residual connections. It has a total of 53 convolution layers [Terven and Cordova-Esparza, 2023]. Aside from a broader architecture, multi-scale predictions, or forecasts at several grid sizes, are an important aspect of YOLOv3. This aided in obtaining better detailed boxes and considerably improved the prediction of small items, which was one of YOLO's key flaws in previous versions. [Terven and Cordova-Esparza, 2023].

Object detector architecture began to be defined in three components at this time: the backbone, the neck, and the head. Figure 2.5 depicts a high-level schematic of the backbone, neck, and head. The backbone is in charge of retrieving meaningful information from the input image.

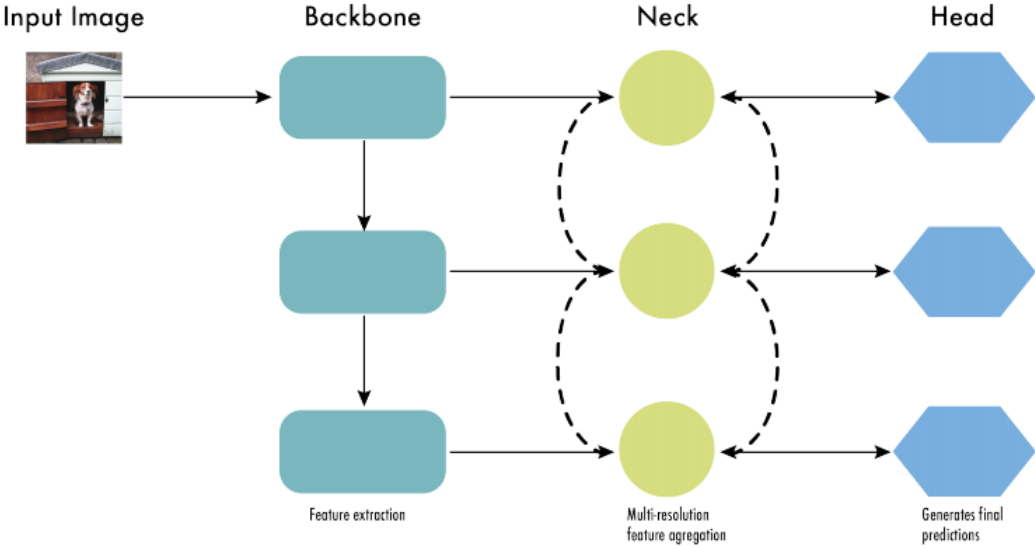


Figure 2.5: YOLOv3 Multi-scale detection architecture [Terven and Cordova-Esparza, 2023].

A CNN trained on a large-scale image classification task is frequently used. Lower-level features (edges and textures) are extracted in the earlier levels, whereas higher-level features (object pieces and semantic information) are extracted in the deeper layers. The neck is a connecting piece that connects the backbone to the head. It combines and refines the features retrieved by the backbone, frequently focused on improving spatial and semantic information at various scales. To increase feature representation, the neck may comprise extra convolution layers. The head is the final component of an object detector. It is responsible for formulating predictions based on the backbone and neck’s attributes. It is often composed of one or more task-specific subnetworks that conduct categorization, localization, instance segmentation and pose estimation. The features provided by the neck are processed by the head, which generates predictions for each object candidate. Finally, a post-processing step, such as non-maximum suppression (NMS), removes overlapping predictions and keeps only the most confident detections [Terven and Cordova-Esparza, 2023].

- YOLOv4: There was no new YOLO version until 2020, when Bochkovskiy et al. published the paper for YOLOv4 in ArXiv [Bochkovskiy et al., 2020]. The authors explored numerous backbone architectures in their experimentation. The objective of this iteration was to maintain real-time detection speed while achieving state-of-the-art per-

formance. YOLOv4 represents a significant advancement in real-time object detection, aiming to overcome the limitations of previous YOLO versions like YOLOv3 and other object detection models. Unlike conventional CNN based object detectors, YOLOv4 is not only suitable for recommendation systems but also for standalone process management and reducing human input. Its compatibility with conventional graphics processing units (GPUs) allows for widespread adoption at an affordable cost, and it is optimized to operate in real-time on a single GPU during both training and inference [Ultralytics, 2023].

- YOLOv5: Glen Jocher, creator and CEO of Ultralytics, launched YOLOv5 a few months following YOLOv4 in 2020. Many of the enhancements discussed in the YOLOv4 are used. It uses a stridden convolution layer and a large window size to reduce memory and computational expenses [Terven and Cordova-Esparza, 2023].
- YOLOv6: This model pioneers several significant enhancements to its architecture and training methodology, featuring the incorporation of a Bi-directional Concatenation module, an anchor-aided training strategy, and refined backbone and neck designs to achieve unparalleled accuracy on the COCO dataset [Ultralytics, 2023].
- YOLOv7: It introduces a novel re-parameterized model, designed to be adaptable across layers in various networks, with a focus on gradient propagation pathways. In addressing the challenge of training models with multiple output layers, YOLOv7 tackles the question of assigning dynamic targets to outputs from different branches. To overcome this, it introduces a novel label assignment approach termed "coarse-to-fine lead guided label assignment." Additionally, YOLOv7 proposes innovative methods such as "extend" and "compound scaling" to enhance the efficiency of real-time object detection systems, effectively leveraging parameters and computational resources. With regards to efficiency, YOLOv7's approach demonstrates significant gains, achieving approximately 40% reduction in parameters and 50% reduction in computation. Furthermore, it offers faster inference speeds and higher detection accuracy [Ultralytics, 2023].
- YOLOv8: Released in January 2023 by Ultralytics, the same company behind the development of YOLOv5, YOLOv8 introduced five scaled versions: YOLOv8n (nano), YOLOv8s (small), YOLOv8m (medium), YOLOv8l (large), and YOLOv8x. Utilizing a similar backbone as YOLOv5, YOLOv8 adopts an anchor-free model with a decoupled head, allowing for independent processing of objectness, classification, and regression tasks. Leveraging state-of-the-art backbone and neck architectures, YOLOv8 enhances feature extraction and object detection performance. With a variety of pre-trained

models available, YOLOv8 caters to diverse tasks and performance requirements, simplifying the selection process for users seeking the ideal model for their specific needs [Ultralytics, 2023].

This design allows each branch to focus on its task and improves the model's overall accuracy. YOLOv8 uses IoU loss function for bounding box loss and binary cross-entropy for classification loss. These losses have improved object detection performance, particularly when dealing with smaller objects.

YOLOv8 has a number of advantages for object recognition. It makes fewer predictions, which speeds up inference time without sacrificing accuracy. Furthermore, YOLOv8 adds a quicker NMS procedure, which effectively removes extraneous bounding boxes. These developments help the YOLOv8 model run more quickly and effectively [Terven and Cordova-Esparza, 2023].

Despite its advantages, YOLOv8 has numerous drawbacks. Due to its anchor-free design, one disadvantage is that it might have trouble recognizing small objects. It may result in false positives in busy settings, which is another drawback [Terven and Cordova-Esparza, 2023].

Technology for object detection has advanced significantly as YOLO has evolved. From the ground-breaking real-time detection capabilities of YOLOv1 to the lightweight and mobile-friendly design of YOLOv8, each iteration has improved upon the accuracy and addressed the shortcomings of its forerunners, with performance metrics playing a crucial role in objectively evaluating these advancements and guiding further enhancements.

## Object Detection Metrics

Performance metrics are crucial in evaluating object detection models as they provide quantitative measures of model effectiveness and efficiency, enabling researchers to objectively compare different models, track progress in the field, identify weaknesses, and guide hyperparameter tuning. These metrics also help in understanding trade-offs between accuracy, speed, and resource usage, facilitating informed decisions for model deployment. Additionally, performance metrics serve as benchmarks for communicating research findings and ensuring quality assurance in real-world applications, contributing to the continual improvement and optimization of object detection algorithms.

Confusion matrices help in computing performance metrics by providing a detailed breakdown of true positive, true negative, false positive, and false negative predictions, enabling the calculation of metrics such as precision, recall and accuracy which are essential for evaluating the effectiveness and robustness of object detection models. To construct a Confusion Matrix for a specific class, we need to consider four key components: True Positive ( $TP$ )

which occurs when the model correctly predicts the class, aligning with the actual ground truth label; True Negative ( $TN$ ) when the model correctly predicts the absence of the class, and this aligns with the ground truth label as well; False Positive ( $FP$ ) when the model incorrectly predicts the presence of the class, while the ground truth label indicates its absence; False Negative ( $FN$ ) when the model incorrectly predicts the absence of the class, while the ground truth label indicates its presence. Confusion matrices play a crucial role by offering a detailed breakdown of TP, TN, FP, and FN predictions, which are fundamental for assessing metrics such as accuracy, precision, and recall, essential for evaluating the robustness and reliability of these models.

Accuracy refers to the proportion of correctly classified instances among all instances.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}. \quad (2.6)$$

This an important metric because it provides a comprehensive understanding of the model's overall correctness in predicting different classes. By considering both true positives and true negatives, accuracy offers a balanced assessment of the model's performance across all classes. However, accuracy alone may not be sufficient when classes are imbalanced or when different types of errors have varying degrees of importance. Therefore, while accuracy is valuable, it is essential to consider additional metrics such as precision and recall.

Precision refers to the ratio of correctly predicted positive instances ( $TP$ ) to the total number of positive predictions made by the model, providing a measure of the model's ability to accurately identify relevant objects without falsely including irrelevant ones,

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{Total\ Predicted\ Positive}. \quad (2.7)$$

This measure becomes crucial when misidentifying positive cases has significant consequences. Precision can be treated as the accuracy of your positive predictions. It is especially important when false positives are costly, helping to focus on the true positives that matter most.

Recall, also known as sensitivity or true positive rate, measures the proportion of actual positive cases that were correctly identified by the model, indicating its ability to correctly detect all relevant instances of a class within a dataset.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{Total\ Actual\ Positive}. \quad (2.8)$$

A model with a high recall catches all positives, but might also include many false positives. In some cases, aggressively striving for high recall can lead to overfitting, where the model performs well on the training data but fails to generalize to new unseen data. This can result



in unreliable performance in real-world applications. Precision and recall are frequently traded off, for example, increasing the number of detected objects (higher recall) can result in more false positives (lower precision).

The Mean Average Precision  $mAP$ , is a popular metric for assessing the effectiveness of object identification models. It calculates the average precision across all categories and provides a single figure with which to compare alternative models. The  $mAP$  metric is built on precision-recall metrics, and it handles numerous object categories while defining a positive prediction:

$$mAP = \frac{1}{C} \sum_{k=1}^C AP_k. \tag{2.9}$$

Object detection strives to accurately pinpoint items in images by estimating bounding boxes. To assess the quality of the predicted bounding boxes, the Intersection over Union ( $IoU$ ) measure is used.  $IoU$  is a metric used to evaluate the accuracy of object detection models by measuring the overlap between the predicted bounding box and the ground truth bounding box, calculated as the ratio of the intersection area to the union area of the two boxes, as shown in Figure 2.6.

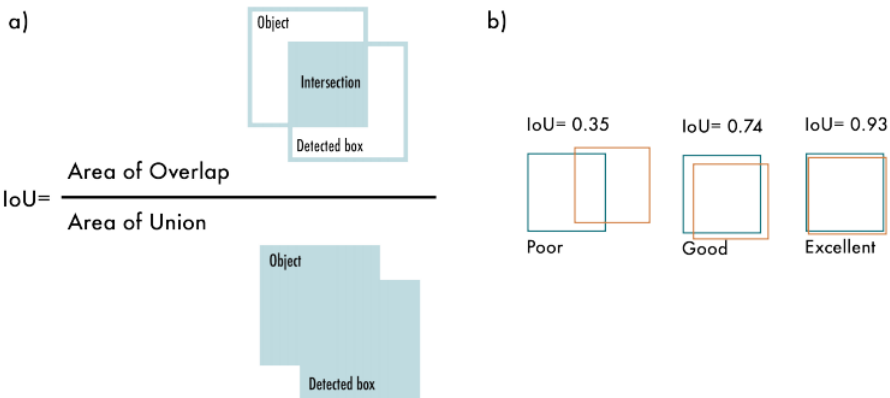


Figure 2.6: a) IoU Formula; b) 3 examples of IoU values for different box predictions [Terven and Cordova-Esparza, 2023].

### 2.3 Transfer Learning

A significant area of machine learning and AI research is transfer learning. Its major objective is to use knowledge from one field or work to enhance performance in another one. Transfer learning has drawn a lot of interest since it can solve the issue of little labeled data and lower the computational expense of developing deep learning models. It has uses in many fields, such as recommendation systems, natural language processing, and computer vision. Transfer

learning's importance rests in its capacity to improve the generalization and effectiveness of machine learning models, enabling them to function well even with constrained resources.

### Fundamentals

Fundamentally, transfer learning is the practice of using the information and skills acquired from one activity or area to improve performance on a related or even unrelated task, as illustrated in Figure 2.7. Transfer learning essentially embodies the core ideas of reuse and adaptability. It recognizes that the information gained from resolving one issue can be extremely helpful when tackling an unrelated but yet related one. This idea bridges the gap between traditional machine learning and AI systems that can display some level of "intelligence" or adaptability taking into account human capacity to generalize and apply previously learned knowledge to new, unknown contexts. Transfer learning holds the potential of efficiency, scalability, and improved performance in a variety of AI applications when used effectively [Pan and Yang, 2009].

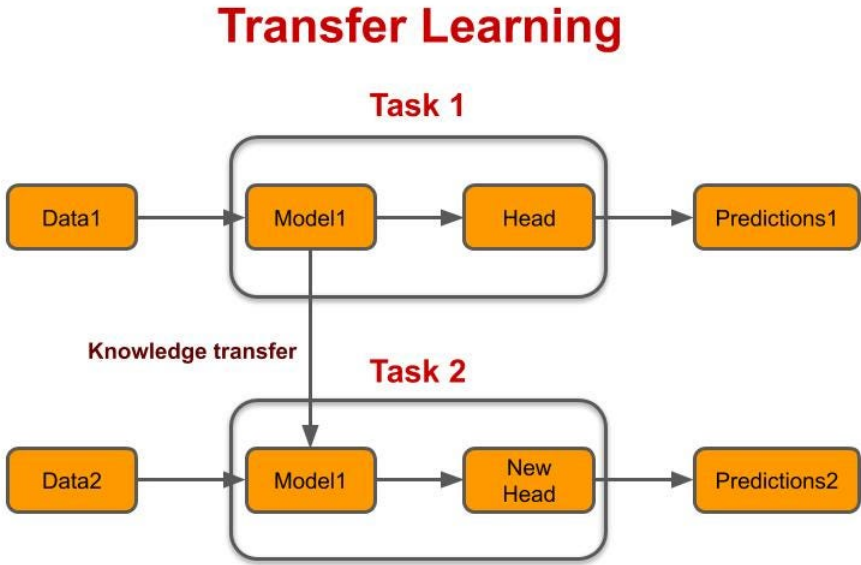


Figure 2.7: Transfer Learning between tasks [NLP, 2019].

Transfer learning includes a variety of tactics, each of which is catered to particular situations and difficulties. Domain adaptation stands out among them as a noteworthy method. It handles the situation where the source domain, from which information is transferred, and the target domain, in which the model must perform effectively, diverge greatly. Another popular technique, fine-tuning, enables pre-trained models to be modified or improved on a specified job, concentrating their knowledge on certain applications. On the other hand,

multi-task learning incorporates a model that is trained to do several tasks concurrently, where the shared information between different tasks improves performance. A cornerstone in the effort to create more effective and capable machine learning models, these various transfer learning strategies enable AI systems to adapt and generalize [Ruder, 2017].

## Approaches

Transfer learning provides a flexible range of methods for applying knowledge from one area to new tasks. These methods improve model performance while simultaneously speeding up the training process and conserving computational resources.

The foundation of transfer learning is pre-trained models, which reflect neural networks that have been previously trained on massive datasets for certain tasks. When applied to jobs outside of their original scope, these models' performance can be considerably improved by utilizing the features they learn [Simonyan and Zisserman, 2014, He et al., 2016, Devlin et al., 2018]. This method takes advantage of the notion that lower-level features, such as edges or textures, frequently have universal value across a variety of domains, expediting learning and fine-tuning for new goals.

For higher level features, fine tuning, which enables practitioners to modify previously learned models to fit particular tasks, is a good approach. The earliest layers, which have already recorded general traits, are kept in this process, and the subsequent layers are either replaced or modified to fit the new task [Yosinski et al., 2014]. The difficult balancing act of fine-tuning involves taking into account the specifics of the target domain while keeping knowledge from the source domain. When the target task and the original job are comparable, like in picture classification tasks where the object categories overlap, it is very effective [Sharif Razavian et al., 2014].

A different method within transfer learning is feature extraction, which enables the use of lower-level characteristics derived from previously trained models as inputs for new tasks. Researchers can avoid having to retrain the entire model by using these extracted characteristics, which reduces computing overhead [Donahue et al., 2014]. When computational resources are scarce or the target job only requires a portion of the knowledge recorded in the pre-trained model, this strategy is useful.

Techniques for domain adaptation are essential for overcoming the difficulty of transferring knowledge between several domains. Domain adaptation approaches try to align these distributions when the source and target domains have different data distributions, which improves knowledge transfer [Ganin et al., 2016]. By incorporating a domain classifier that promotes the model to create features that are domain-agnostic and facilitate greater performance on the target domain, techniques like adversarial domain adaptation have become

more popular [Tzeng et al., 2017].

## Challenges and Considerations

When the data distribution in the source and destination domains differs, transfer learning frequently faces the significant obstacle of domain shift. When applied to target data with differing features, the pre-trained models may have trouble. The prevention of domain shift is essential for transfer learning to be successful. Domain adaptation techniques, which align the feature distributions between domains, have become effective approaches [Dimitriou, 2021]. Furthermore, fine-tuning the pre-trained model on a constrained target dataset aids in its adaptation to the specifics of the new domain [Shazeer, 2020]. These techniques are essential for bridging the divide between the source and destination domains and enabling more efficient knowledge transfer.

Robust data preprocessing techniques, such as noise reduction and outlier detection, can increase the dependability of the target data when data quality is an issue. For optimal transfer learning outcomes, it is crucial to strike a balance between data size and quality.

To summarize, transfer learning is more than just a tool, it is a strategic necessity in the dynamic world of deep learning. Its significance stems from its capacity to effectively apply previously acquired information from one field or job to another, bringing about several compelling benefits. Transfer learning is perhaps most notable for its time-saving beacon function, which significantly speeds up the tedious process of training deep neural networks from the beginning. By using pre-trained models as potent starting points, it conserves precious computing resources. Transfer learning has also shown to be a potent ally in the fight against data scarcity.

## 2.4 Pose Estimation

By carefully monitoring critical spots on the human body, Open Pose, a computer vision technology, plays a crucial role in the field of AI in sports by providing a thorough analysis of posture and biomechanics. Open Pose is used by coaches and sports scientists to acquire insightful information about how athletes move, enabling accurate evaluation of technique, posture, and motion patterns. Through the analysis of biomechanical data and in addition to identifying movements in the case of Padel, pose estimation capabilities extend to injury prevention and recovery, assisting trainers in developing individualized plans to reduce risks and hasten recovery.

### Background

Open Pose has created new opportunities for performance monitoring by allowing coaches

to precisely gauge vital stats like joint angles, speed, and power. These developments have revolutionized how athletes practice and get ready for competitions, ultimately improving their performance as a whole. While Open Pose technology has many benefits, it also has certain drawbacks. The need for superior imaging technology to accurately capture body movements is one restriction [Wei et al., 2021]. In real-time sports situations when athletes are moving, this can be difficult. Furthermore, Open Pose could have trouble correctly identifying poses in complicated movements or obscured body components [Badiola-Bengoia and Mendez-Zorrilla, 2021]. Despite these drawbacks, continued research and improvements in AI algorithms keep Open Pose technology's accuracy and robustness in sports training from declining.

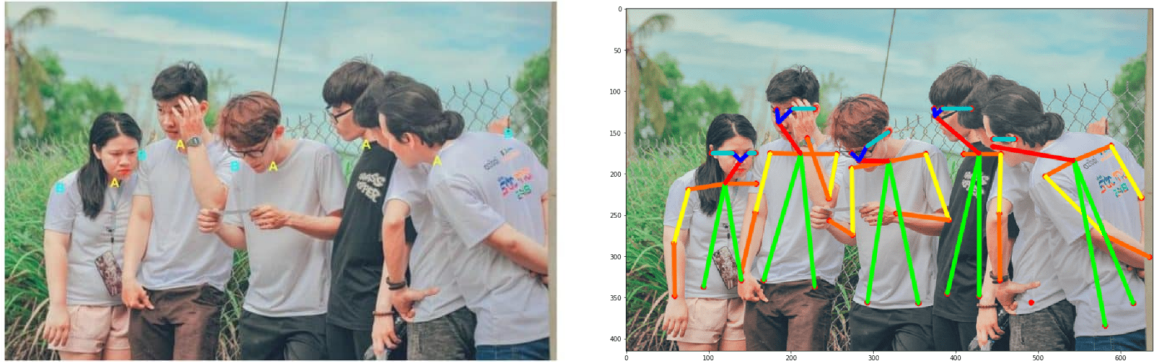
## Methodology

A multi-step process combining computer vision techniques, machine learning algorithms, and real-time analysis is used to deploy Open Pose in sports training. The first step in the process is gathering high-quality data using various image devices, like cameras or depth sensors. Advanced computer vision techniques are then used to process this data in order to precisely recognize and monitor human body motions. The Open stance algorithm estimates a person's stance in real time by combining body part localization and part affinity fields [Cao et al., 2017]. CNN is a popular approach for creating Open Pose and have demonstrated outstanding performance in image identification tests. The CNN learns to recognize and locate important human joints, as shown in Figure 2.8, and limbs after being trained on a sizable collection of annotated images. The whole body pose can then be estimated using this information. Depending on the exact requirements and resources available, Open Pose model can be implemented using a variety of CNN architectures [Newell et al., 2016].



Figure 2.8: Keypoints of the skeleton [\[in OpenCV using OpenPose, 2018\]](#).

Video or depth maps of athletes doing different actions are commonly used in the data-collecting procedure for human pose estimation in sports training. The models are trained using this data as the source of reality. High-resolution cameras are typically used to record the video, which are placed in strategic locations to capture the motions of the athletes from various angles. To measure the separation between the sensor and the subject, depth sensors or structured light cameras can produce depth maps. Key body joint positions must be added to the data once it has been obtained [\[Gkioxari and Malik, 2015\]](#). Each frame of the video or depth map is manually annotated to indicate where joints like the shoulders, elbows, knees, and ankles are located. The model is then trained to recognize and properly predict human poses using this annotated data. Several strategies are used to infer human poses from the gathered data. Based on color, texture, or depth information, these algorithms examine the video frames or depth maps to identify and locate important body joints. After being trained on the annotated data, a CNN can reliably estimate poses since they have learned the correlation between image attributes and body joint locations [\[Pfister et al., 2015\]](#).



(a) Open Pose nose - neck matching points candidates. (b) Open Pose skeleton detection.

Figure 2.9: Open Pose skeleton detection example [in OpenCV using OpenPose, 2018](#).

The 2D joint locations, illustrated in Figure [2.8](#), of the input images are directly regressed using a CNN, as shown in the example of the Figure [2.9](#). Particularly in difficult situations with occlusions or complex movements, this model has demonstrated excellent accuracy in estimating human positions. To understand the intricate connections between image elements and human poses, these AI algorithms and models are trained on enormous datasets that have been annotated. They can properly estimate the positions of bodily joints by evaluating the input data [Wei et al., 2016](#).

In conclusion, the integration of OpenPose model into an AI system for Padel sports represents a significant advancement, by tracking players' movements and gestures on the Padel court in real-time accurately. This not only enables comprehensive analysis of player techniques but also facilitates the identification of the movement itself. Furthermore, after having a movement classification, understanding the player positioning on a Padel court is crucial for executing precise analysis and statistics generation, making it essential to accurately identify the court boundaries to facilitate optimal gameplay and strategic decision-making.

## 2.5 Court Detection

### Canny Edge

Edge detection is a key idea in computer vision that is essential to processing and analyzing images. It entails locating and extracting the edges or borders of items contained inside an image. These margins, which signify notable variations in brightness or hue, reveal important details about the composition and subject matter of the image. Edge detection has value in computer vision because it makes image comprehension, object recognition, and segmentation jobs easier. Computer vision systems can perform object tracking, extract useful characteristics, and enable a variety of applications like scene interpretation, image-

based navigation, and visual inspection by precisely detecting edges. Sports-related tasks like player tracking, shot analysis, and tactical decision-making depend on the precise recognition and localization of the Padel court within an image or video frame. Coaches, players, and analysts can benefit from the automated extraction of pertinent data from video footage thanks to court detection. It makes it possible to recognize court lines, the position of the net, and other court markers, simplifying advanced sports analytics and improving overall game comprehension.

The Canny edge detection algorithm is one of the most well-known algorithms in the enormous body of literature on edge detection methods. John Canny developed the Canny algorithm, which has been widely used because of its accuracy and robustness in recognizing edges. The Gaussian smoothing, gradient computation, NMS, and hysteresis thresholding are some of the crucial elements that make up the Canny algorithm [Canny, 1986]. Together, these actions enable the detection and highlighting of image edges while reducing noise and erroneous detections. The algorithm's performance depends on its capacity to generate narrow, well-connected edge outlines and detect edges properly with low error rates.

### Implementation

The Canny edge detection technique contains numerous crucial steps and is a multi-step procedure [Canny, 1986]. In order to use the Canny edge detection algorithm on images or videos of Padel courts, the first step should be starting by loading the Padel court image or video frame and perform image preprocessing. Then, since edge detection mostly relies on intensity variations, converting the image to grayscale is the next step. Afterwards, comes the application of a Gaussian blur to the grayscale image to lessen noise and minute intensity variations. Subsequently, methods like Sobel filters to determine the gradient of the blurred image are employed [Canny, 1986]. This phase reveals areas with sudden variations in intensity, which frequently coincide with edges. Thereafter, locate regional gradient magnitude maxima at the boundaries. This eliminates edges that are unnecessary. Posteriorly, apply dual thresholding by establishing a high and a low threshold for thresholding. Strong edge pixels are those with gradients above the high threshold, whereas weak edge pixels are those with gradients between the low and high thresholds. Thereupon, extend strong edges and reduce the weak ones, while using edge tracking and connectivity analysis. Then, to create the final edge map, combine the strong and extended weak edges and play back the generated edge map to see the edges that were picked out in the Padel court output image [Canny, 1986]. Figure 2.10 illustrates the application of Canny Edge to an image of the Padel court.





Figure 2.10: Canny Edge application example.

In some instances, further sharpening the edges that have been detected can improve the precision and quality of the outcomes. There are numerous post-processing techniques available as edge smoothing, which uses morphological processes like dilation and erosion to link and smooth out disjointed edges, resulting in a more cohesive representation. Another technique is noise reduction, which uses methods for suppressing isolated noisy pixels that may have been mistaken for edges, such as median or bilateral filtering. Also, the Hough transform can be used to find and represent lines or curves inside of edges that have been recognized. This might help locating particular aspects of the image.

## Hough Lines

The Hough Transform overcomes the constraints of gradient-based edge detectors to enable the extraction of lines from images. The Hough Transform offers a sophisticated answer to the problem of line detection, even in the presence of noise and gaps, by converting the spatial domain of lines in Cartesian coordinates to the parameter space of lines.

The Hough Transform's theoretical underpinnings stem from its capacity to solve the issue of line detection in an image space, permitting the representation of lines in parameter spaces. The Hough Transform is a key technique in computer vision and image analysis that was first presented by Paul Hough in the 1960s for the detection of lines in particle tracks within bubble chamber pictures [Hough, 1962]. The core concept of the Hough Transform is its ability to convert an image's Cartesian representation of lines into a parameter space where lines are expressed as points, making it easier to locate and analyze these lines.

Lines can be represented mathematically in several ways, including the standard slope-intercept form

$$y = mx + b, \quad (2.10)$$

the normal form

$$x\cos(\theta) + y\sin(\theta) = \rho, \quad (2.11)$$

and the parametric form

$$\rho = x\cos(\theta) + y\sin(\theta). \quad (2.12)$$

The parametric form is used by the Hough Transform because it is straightforward and flexible [Duda and Hart, 1972]. The quality of parameter discretization and the thresholding method used to identify significant peaks determine how well the Hough Transform performs [Ballard, 1981].

In conclusion, the theoretical foundation of the Hough Transform has made it possible for it to be widely used in applications ranging from robotics to medical image analysis. It is a flexible tool for computer vision and pattern recognition since it provides a reliable method for identifying not only straight lines but also other shapes that can be specified.

### Improved Hough Transform Techniques

The Probabilistic Hough Transform (PHT) and the Randomized Hough Transform (RHT), two well-known versions, have come to light as significant alternatives to the Standard Hough Transform that offer notable advancements in both accuracy and effectiveness.

By addressing the computational inefficiencies of the Standard Hough Transform, the PHT represents a substantial advancement. By randomly choosing some edge points and aggregating votes just for those points, it drastically decreases the processing time when a subset of the edge points is sufficient to detect the lines of interest. With a high likelihood of finding the desired lines, this method significantly increases efficiency. Due to its ability to achieve a compromise between accuracy and processing speed, the methodology, which was first presented by [Matas et al., 2000], has subsequently become a crucial technique in line identification algorithms.

Another invention that promises gains in precision and effectiveness is the RHT, which deliberately chooses a subset of edge points and uses their data to establish the parameters of the lines. It was first proposed by [Yuen et al., 1990]. The method is suited for real-time applications or situations with constrained computer resources because of this selection process, which lowers the computational complexity. The RHT provides accuracy comparable to the Standard Hough Transform while drastically decreasing the computational load by effectively sampling the parameter space.

These Hough Transform variants include adjustments and improvements that address the drawbacks of the Standard Hough Transform. When processing images with plenty of edges, the Probabilistic Hough Transform accelerates the procedure by concentrating computation on important edge points. The RHT, on the other hand, strikes a balance between accuracy and effectiveness by carefully choosing representative sites for parameter estimation. These methods address a variety of application scenarios and address the problems the original

method had with computational complexity. The trade-offs between efficiency and accuracy are crucial factors to take into account while using these upgraded procedures. The Standard Hough Transform ensures accuracy but frequently comes at the expense of longer computation times. While obtaining a similar level of accuracy, the PHT and RHT thrive in situations when computational effectiveness is crucial. These trade-offs demonstrate the adaptability of the improved Hough Transform variations, allowing to select the most appropriate technique in accordance with their unique needs and limitations.

### Challenges and Strategies

To provide accurate and trustworthy findings, some issues must be resolved when utilizing the Hough Transform to detect lines in images.

Accurate line identification requires addressing the drawbacks and shortcomings of the Hough Transform. The algorithm's susceptibility to noise in the input image is one of the main causes of concern. The accumulation of votes in the Hough space may be wrong because noise can produce erroneous edge points. The detection of lines with particular orientations and positions can also be impacted by errors in parameter selection, such as the granularity of discretization [Duda and Hart, 1972].

To mitigate noise-related problems, it is essential to use noise reduction methods and smoothing filters [Gonzalez and Woods, 2008]. Beyond noise, several intersecting lines in an image can make the detecting process more challenging. [Ballard, 1981] illustrates how intersecting lines might result in several local maxima in the Hough space, which can make peak detection more difficult. To distinguish between overlapping lines and precisely calculate their properties, we need sophisticated techniques. Additionally, choosing the right peak detection thresholds can affect the results. These difficulties emphasize the significance of reliable thresholding methods that can adjust to various image situations.

Post-processing techniques and adaptive thresholding approaches are methods for overcoming these difficulties [Sonka et al., 2014]. Following peak recognition, a geometric connection analysis of the lines found can confirm their existence and eliminate extraneous or erroneous lines. Adaptive thresholding enables the threshold values to be changed by local picture features. This flexibility increases the algorithm's resistance to changes in noise intensity and image quality [Gonzalez and Woods, 2008].

To sum up, the Hough Line detection method is a key component of computer vision applications. Due to its skill at identifying lines in images, it has enabled vital applications such as shape identification, text extraction from documents, and lane detection in autonomous vehicles. It cannot be stressed how important the Hough Transform is to image processing, analysis, and machine vision as it continues to influence the field of computer vision

applications [Sonka et al., 2014]. The Hough Transform continues to make significant contributions that highlight its significance in computer vision and indicate its lasting influence on upcoming technological developments.

## 2.6 Ball Detection

A crucial task in computer vision is object detection, which enables computers to recognize and locate things in pictures or video frames. In this area, Kernelized Correlation Filters (KCF) have become a notable approach, providing creative answers to the problems of real-time object tracking and recognition. To build reliable and effective object detection models, KCF combines kernel approaches with the capability of correlation filtering [Bolme et al., 2010].

KCF has become a well-known and effective real-time tracking technique in recent years. KCF stands noteworthy for its capacity to carry out reliable tracking even in difficult conditions such as occlusion, scale variation, and sudden changes in motion.

### KCF

Kernel-based correlation filtering, a potent method that enables reliable and effective object tracking and detection, is at the heart of KCF. This method is based on the idea of kernel functions, which take input data and transform it into a higher-dimensional space where linear correlations between features can be successfully exploited. This modification is essential to KCF's object identification method since it makes it easier to track and localize objects in images and videos. KCF is especially well-suited for dynamic real-world applications because it can adapt to changes in object appearance and scale by cross-correlating a template patch with the target region using kernel functions [Bolme et al., 2010].

KCF stands out among other object identification and tracking methods thanks to several compelling features it provides. Its resistance to changes in appearance may be one of its most notable benefits. KCF excels in situations when object appearances fluctuate as a result of elements like lighting, partial occlusions, or shifting viewpoints, as shown in Figure 2.11. The tracked object will always be recognizable thanks to KCF's usage of kernel-based correlation filters, which can successfully mimic these appearance fluctuations. Additionally, KCF shows astounding efficiency, especially in real-time tracking applications. Due to its fast processing speed, it can process video feeds in real time while maintaining high tracking accuracy, which is essential in applications like surveillance, robotics, and autonomous cars [Henriques et al., 2014, Bolme et al., 2010].

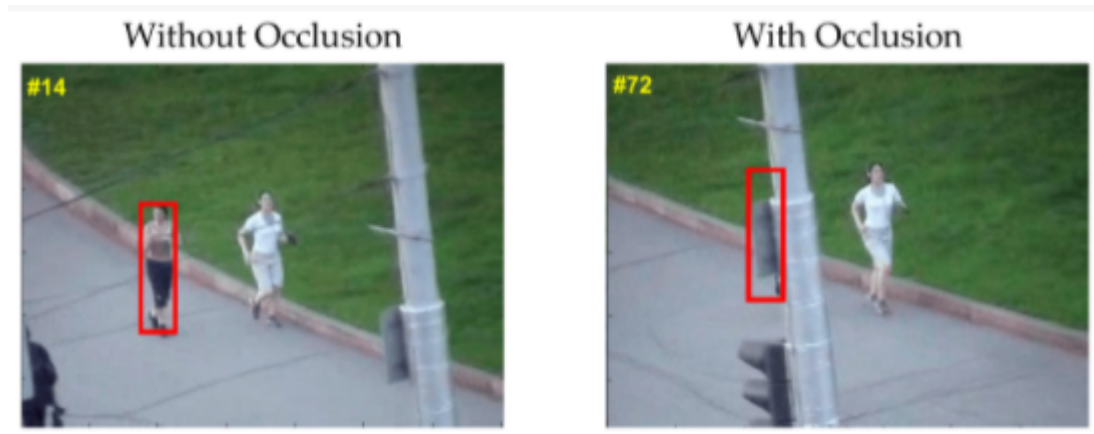


Figure 2.11: KCF occlusion example [Mehmood et al., 2021].

The pipeline used by KCF, which includes many stages of object identification, is the foundation of the system. The initialization step, when the object of interest is chosen or given as a reference region, is the core of this process. KCF makes use of kernel-based correlation filtering to examine the visual features in this reference region and identify the defining traits of the item. In essence, during this stage, a filter that captures the object's look is trained, making it ready for tracking to come later. Once the filter has been initialized, the tracking phase begins, during which all of the image or video frames are convolved with the filter. Real-time tracking is made possible by the peak response of this convolution, which identifies the object's position in each frame. Additionally, KCF has localization features that enable it to precisely determine the bounding box coordinates of the object [Henriques et al., 2014, Bolme et al., 2010].

KCF stands out as a unique approach with its own set of strengths and weaknesses when compared to other widely used techniques in the constantly changing field of object recognition. When contrasting KCF with well-known object identification techniques like YOLO, a clear contrast is revealed. One standout characteristic is its deftness in handling items and scenes that are constantly changing. Although KCF can be modified for detection tasks, its primary focus is tracking, therefore in dense detection cases where precise object localization is crucial, it might not equal the precision of YOLO. These additional methods were developed specifically to detect objects, frequently by utilizing deep learning architectures. For instance, YOLO has remarkable object identification speed and accuracy, but it might fall short of KCF's tracking capability in real-time applications. KCF's main advantages for tracking are its effectiveness and adaptability, whereas YOLO is superior for complex object recognition tasks [Redmon et al., 2016, Ren et al., 2015, Wu et al., 2017].

In conclusion, KCF is a flexible method that excels in object tracking and detection by utilizing kernel-based correlation filtering. Furthermore, KCF is a useful tool for applications

that require low-latency object detection because of its high computational efficiency.

## 3 Research Methodology

### 3.1 Research Design

Several methods, including pose estimation, image annotation and fine-tuning of the YOLO model, were employed to construct an AI system that understands key points and moments of the Padel game and is capable of generating statistics by itself.

Good data quality is one of the core components of an accurate and trustworthy AI system. Without adequate data, models rely on incorrect patterns to categorize what is required, which produces incorrect statistics.

After the frames were gathered and arranged according to movement, the order of the procedures is the following, as illustrated in Figure 3.1. Pose estimation was carried out gradually for each player on the Padel court. To more accurately determine which movement is conducted in each frame, the movement classifier model makes use of the estimation of the body skeleton as a feature. Following this, each frame that had a pose estimation was labeled.

The obtained frames and respective labels were organized in order to be used by YOLO to perform fine tuning with the new movement classes. The data is separated into three groups: train, test and validation. The model can be trained using a balanced sample of the data when the data is structured as described above. Data is then prepared to feed the YOLO model.

The presence of the Padel court boundaries is still another crucial aspect of the procedure. The frames must be free of distortion to distinguish the field's lines, and, since the camera used at Quinta do Padel contains distortion, this needs to be taken into consideration. The Padel court's lines and bounds are then determined using Canny Edge and Hough Lines approaches.

The positions of the ball during the game must be saved, together with its trajectory, to produce statistics about important points of the game.

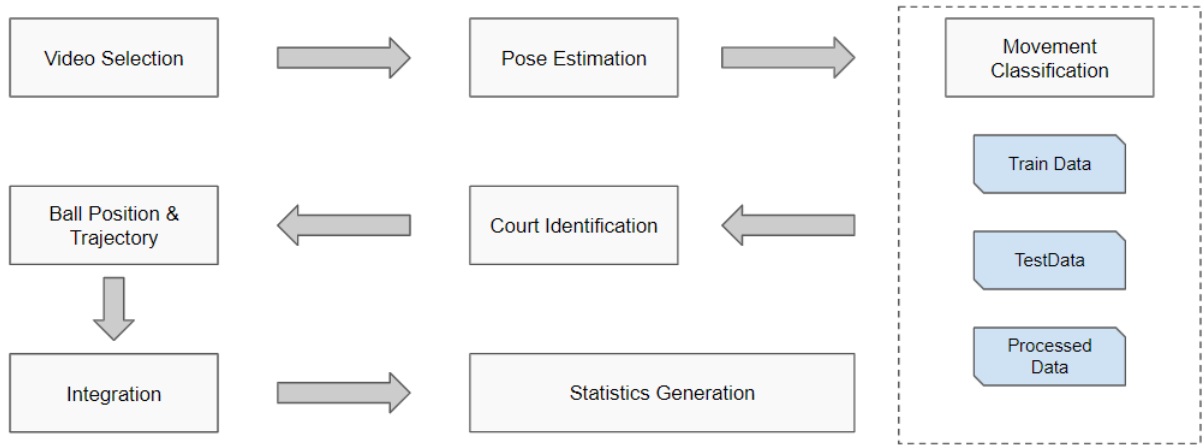


Figure 3.1: Methodology procedures' architecture.

The aforementioned procedures give the system access to the movement identified in each frame in relation to the court and ball coordinates. This enables the system to provide statistics like movement with more points collected, the most accurate player and others by creating a table with the game's timeline. All of these factors work together to create a system that enhances performance monitoring and Padel statistics creation.

## 3.2 Methodology

The methodological part of this master's dissertation takes center stage in the quest to improve performance and strategic decision-making in the world of Padel. The foundation of our investigation into how to apply AI principles to the realm of Padel is provided by this chapter. The methodical process used to design, create, and assess AI-driven solutions for Padel game analysis and player support will be covered in detail in the pages that follow. To prepare for a thorough analysis of how AI can change the game of Padel, this methodology section will carefully lay out the research strategy, data gathering, AI model selection, and evaluation criteria.

### 3.2.1 Movement Classification

#### Data Collection

One of the most important components of an accurate and trustworthy AI system is high data quality. In the absence of proper data, models rely on inaccurate patterns to identify what is necessary, resulting in incorrect statistics. Therefore, choosing quality data to feed the models was a point to consider.

For this, 4 players were chosen to play at Quinta do Padel's court following specific rules



so that the selection of the movements to send to the model was made easier. For this selection, it was decided that for every 5 minutes one of the teams would only perform one specific movement. This way, when selecting frames for a specific movement, the parts of the video where it was performed were already known.

Quinta do Padel's court has two cameras, each one is disposed in each part of the field, as shown in Figure 3.2. Since two cameras are available, every movement was recorded twice, once for each side of the court and consequently for different players.

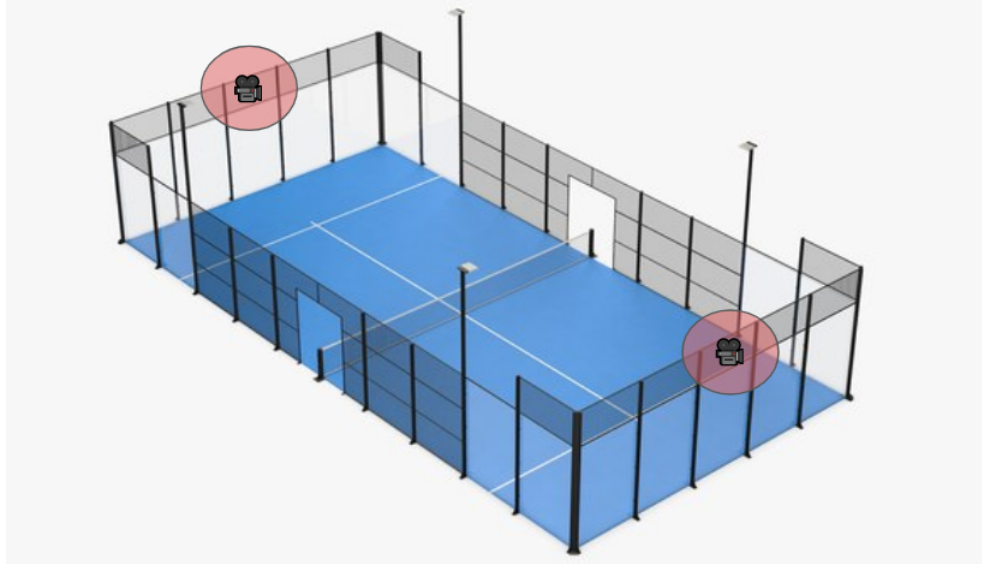


Figure 3.2: Cameras distribution in Quinta do Padel's court.

Having the same movement made by different players in different angles is going to be extremely positive to help the model learning the movement's patterns more accurately. The entire recorded video on this session was then separated by movement and camera.

For this specific work and for a matter of investigation, the focus was only on two movements, Backhand and Forehand.

### Data preprocessing

A preliminary method for training the movement categorization model, without using a skeleton, was developed. The model had difficulty distinguishing between some game-related irrelevant body motions and the precise movement that executed a hit on the ball, hence the results were not as good as expected.

Figure 3.3 shows the difference between the images that were first used and the images containing the pose estimation. The effectiveness of a movement classification model is greatly improved by the precise posture estimation of players.



(a) Frame without skeleton.



(b) Frame with skeleton.

Figure 3.3: Example of application of Pose Estimation to Quinta do Padel court game.

The system receives a precise depiction of the players' body locations and movements on the court from accurate pose estimation. This knowledge is essential for identifying minute

variations in player behavior, such as various stroke types, footwork patterns, and player interactions. The movement classification model can distinguish between these subtle moves by accurately tracking player stances, which eventually enhances the model's capacity to decipher and categorize the players' on-court actions.

To better comprehend the distinction between irrelevant and relevant movements, pose estimation was added to the frames as a new feature of the model, as exemplified in Figure 3.3b. Open Pose approach was used to apply pose estimation on the players to each frame of the video to have the players' skeletons in the frames. Upon data collection, the subsequent procedure entails labeling the data to facilitate the model's understanding of the distinctive characteristics associated with each movement.

### **Labeling**

Image labels are important for several reasons, one of which is that supervised learning, the process through which an AI model learns patterns and features related to several classes or categories, is made possible. The model may develop relationships between visual traits and the matching class labels through labeled samples, basically learning what each element looks like. Without image labels, the model is unable to make an appropriate distinction between these elements. The model can effectively generalize its knowledge from labeled examples to correctly categorize new, unlabeled images seen during gaming since image labels give it a reference point.

Identifying which movement the player is making is crucial in the context of this Padel game investigation. To annotate the player who is executing the movement and label the movement a tool called CVAT was used. Each frame was given a class that was chosen along with the appropriate position skeleton, as shown in Figure 3.4.



Figure 3.4: Example of a CVAT annotation of a Forehand movement.

Following the collection and labeling of data, the subsequent step entails providing it as input to the utilized model, which is going to provide the classification of the movements.

## YOLO

The choices about the base structure and architecture is crucial in the effort to create a model that can recognize objects and anticipate Padel movements, shown in Figure 3.5. Due to its powerful object identification capabilities and versatility for unique applications, YOLOv8 stands out as a desirable candidate.

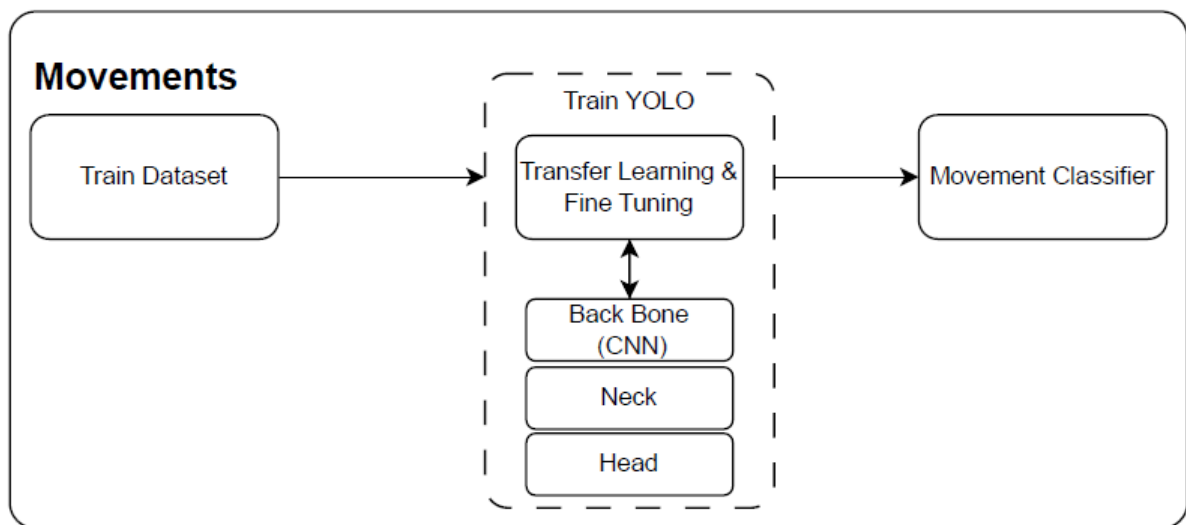


Figure 3.5: Movement classifier structure.

The YOLOv8 model can be initiated with pre-trained weights from massive datasets, thereby enhancing its capacity for object recognition and detection in this domain through transfer learning, which imbues the model with prior knowledge encompassing a diverse range of entities.

It needs to be fine-tuned for Padel object detection in the following stage. The process of fine-tuning is how the model learns to modify its feature representations and weights to the unique properties of the Padel movement dataset.

In order to assess the precision of bounding box predictions, IoU is used. It measures the amount of overlap between an object's ground truth bounding box and the expected bounding box for that object within an image. The amount of overlap between the two bounding boxes is divided by the size of their union to determine the IoU,

$$IoU = \frac{\text{Area of Intersection}}{\text{Area of Union}}.$$

The resulting IoU value is a percentage-based indicator of how closely the projected bounding box matches the actual object's position and dimensions. A better match is indicated by a greater IoU.

In addition to employing this metric for assessing the model's proficiency in projecting a bounding box, it is imperative to assess the model's performance in predicting the corresponding label associated with its bounding box, indicative of the movement executed within that frame. YOLO uses the Cross Entropy loss function, Equation (2.1). This loss function is intended to maximize the model's capacity to correctly categorize the movements.

Beyond the movements, the Padel court needs to be identified so that the position of the players can be then compared to the field.

### 3.2.2 Padel Court Identification

The examination of Padel games using AI shows that the detection of lines inside a Padel court is a vital and important component. In the first place, it delineates the playing field, defining player limits and creating rules for the scoring procedure. Second, the lines are crucial for enforcing the rules because they specify where the ball must land during play and where it should be served. Therefore, it is crucial to correctly identify these lines to follow the ball's trajectory, and players' movements. Figure 3.6 illustrates the strategy used to identify the court boundaries.

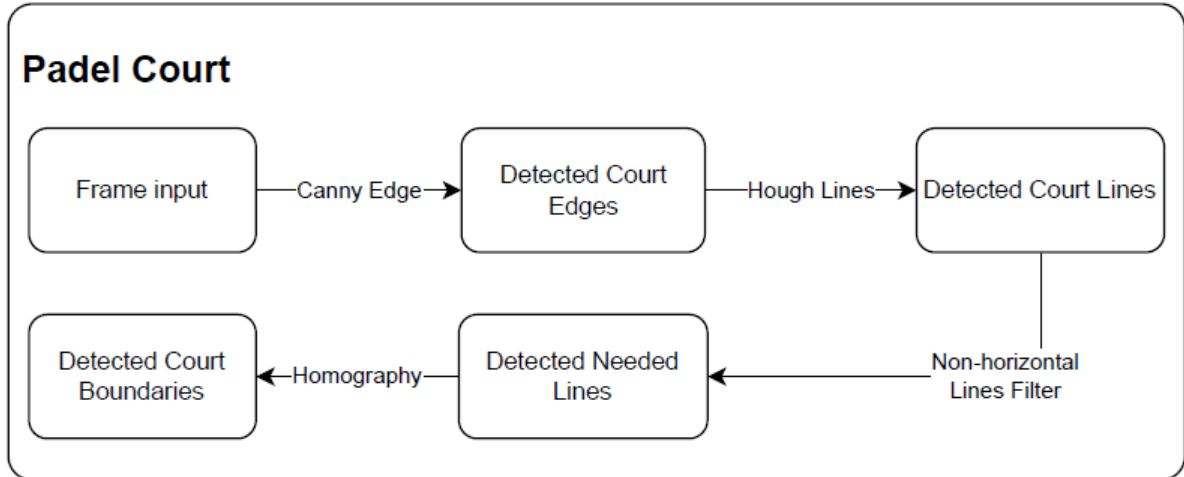


Figure 3.6: Court boundaries identification structure.

Court boundaries detection acts as the stage on which later analyses are constructed. This covers player positioning, ball tracking, and shot trajectory prediction, all of which largely rely on having a perfect understanding of where the lines are. Without this vital data, the automatic analysis of the game dynamics in Padel, such as shot accuracy, player performance metrics, and tactical insights, can be threatened.

A frame of an empty court was chosen as a reference for court detection to identify the Padel court boundaries, as shown in Figure [3.7](#).



Figure 3.7: Frame of the empty court.

This reference image is important since it depicts a clear, unhindered perspective of the

court's layout that is devoid of any players or equipment.

### Canny Edge

The frame is converted into a grayscale image using a conversion process. Reducing the image to a single grayscale channel works well for highlighting the lines on the Padel court, which are the most important visual components. Additionally, grayscale conversion makes it easier to eliminate excessive noise and unwanted elements that could be present in color photographs. This noise reduction is especially important when working with low-quality video or under difficult lighting conditions, which are both common in Padel court situations and it is also helpful to reduce the complexity of the image making it easier to process.

The Canny edge detection algorithm is then applied to the frame to find image edges, as represented in Figure 3.8. In this procedure, the background image is transformed into a binary representation, where pixels that are part of an edge are given a binary value of 255, which represents the color white, and pixels that are not part of an edge are given a binary value of 0, which represents black.

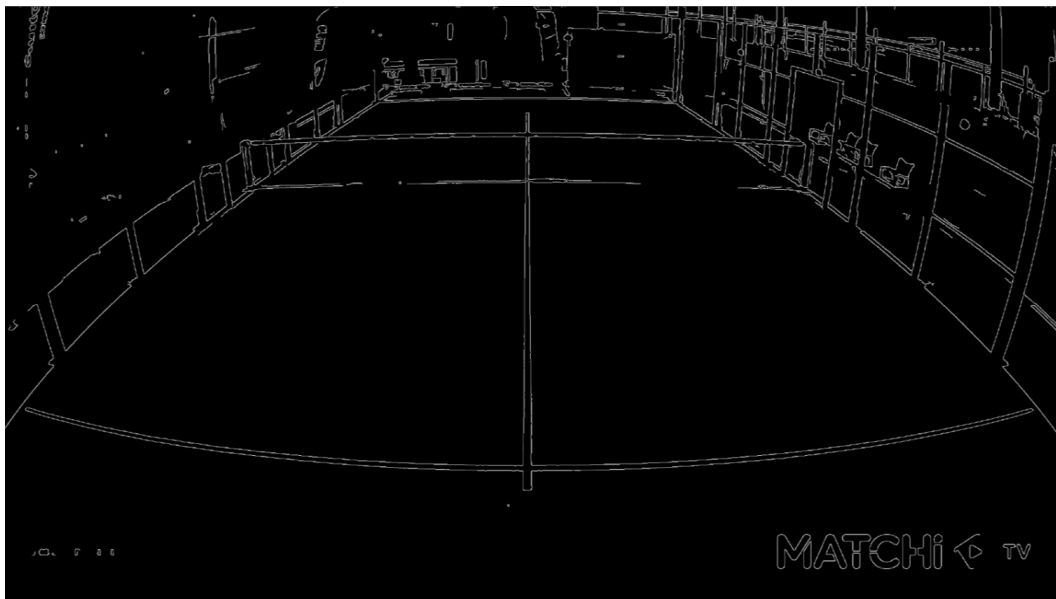


Figure 3.8: Frame of the court after applying the canny edge detector.

The Canny edge detector uses a threshold range to identify which pixels are edges, with a lower threshold of 50 and an upper threshold of 100. The white pixels are subjected to a dilation operation after the image has been translated to binary representation and all of the edges have been located. This dilation improves their visibility, makes them more recognizable, and makes them simpler to find in further study, as shown in Figure 3.9.

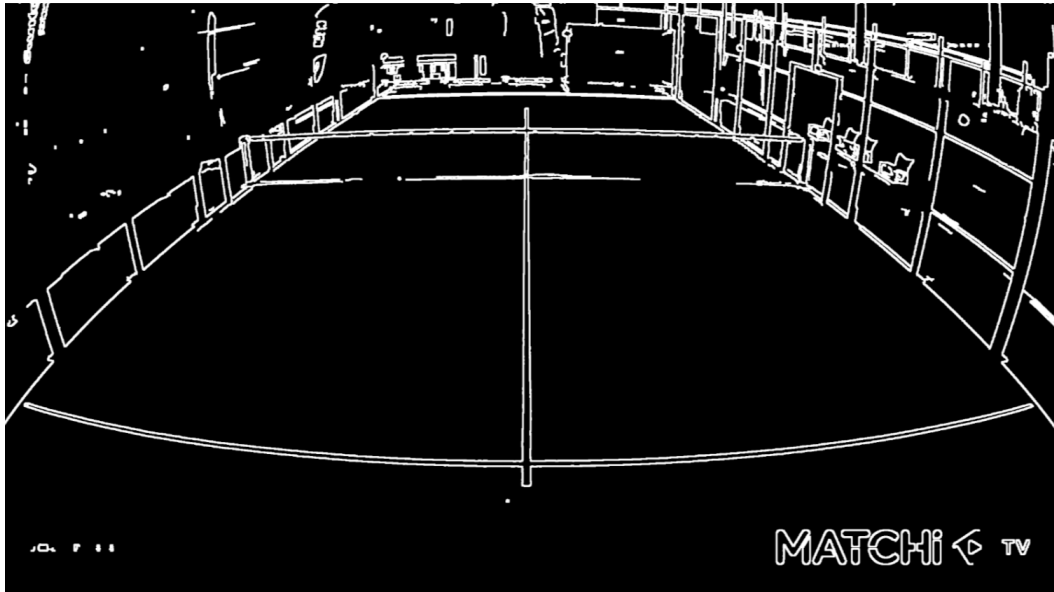


Figure 3.9: Dilated Result of the canny edge detection.

### Hough Lines

To find all relevant lines in the binary Canny image, a probabilistic Hough lines transform algorithm is used, as represented in Figure [3.10](#).

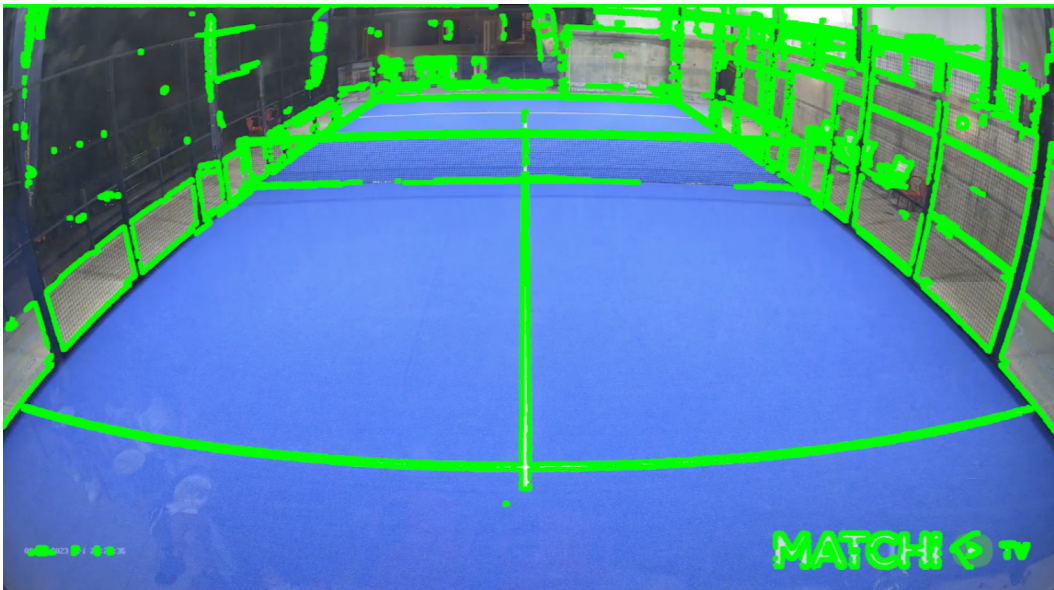


Figure 3.10: Detected court lines.

Vertical lines and horizontal lines are the two separate groups created from the lines produced by the probabilistic Hough lines algorithm. This classification is essential since only horizontal lines contain the important areas of interest, whereas vertical lines are unnecessary



and are therefore not included. Both of each line's ends,  $(x_1, x_2)$  and  $(y_1, y_2)$ , are evaluated as part of the sorting process. The line is referred to as horizontal if the distance along the x-axis difference is greater than twice the distance along the y-axis difference, according to the following condition:

$$| (x_2 - x_1) | > 2 \times | (y_2 - y_1) | .$$

In contrast, the line is classified as vertical if this requirement is not met. It is crucial to combine these smaller lines that represent the same line segment since the probabilistic Hough lines method frequently finds many smaller lines that together represent a bigger line within the frame. To enable the accurate identification of the major endpoints of the lines, this consolidation phase is essential. An effective filtering process is used to remove unnecessary data before merging all of the lines. Based on a length threshold of 100 pixels, this filtering process removes shorter, irrelevant lines, streamlining the dataset and getting rid of superfluous data. A pairwise comparison strategy is used to consolidate horizontal lines, where two lines are analyzed at once and combined based on their pixel locations along the y-axis. The lines are merged if the difference in their y-coordinates is less than 10 pixels, indicating that they can be viewed as parts of a single line, according to the following condition:

$$| (y_2 - y_1) | > 10 .$$

The output of the above conditions met together can be seen in Figure [3.11](#).

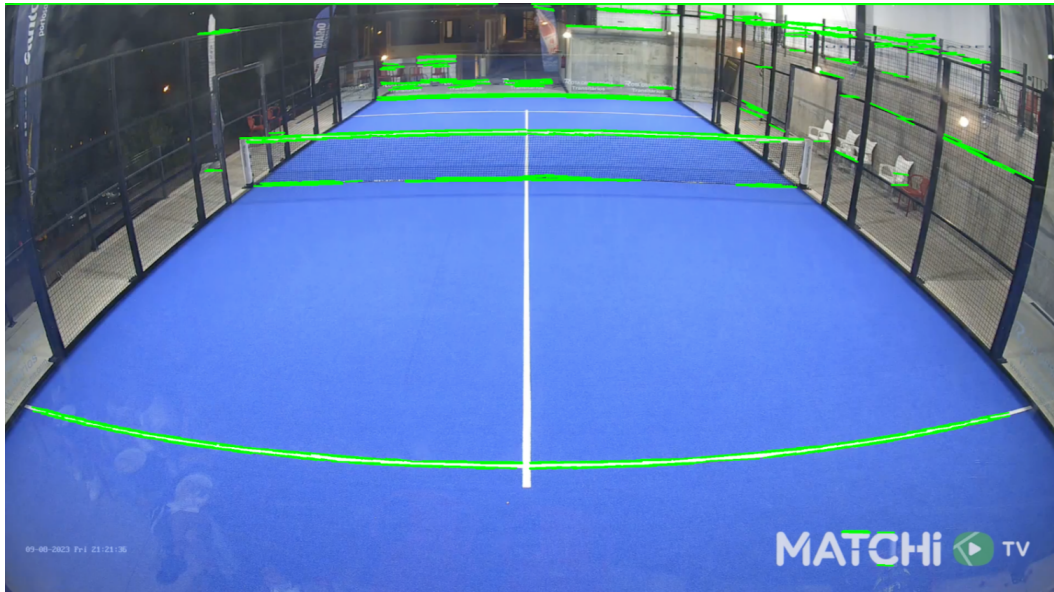


Figure 3.11: Detected court horizontal lines.

All relevant lines are anticipated to exceed this pixel length criterion after merging, thus

any lines with a length of less than 200 pixels are disregarded. Based on their y-values, the remaining collection of lines is then further refined. The array is shifted so that the line with the lowest y-value is at the front and the line with the highest y-value is at the back. The first and last lines in the array are selected as the four critical locations necessary for computing the homography transformation thanks to these operations.

## Homography

Homography, also known as a perspective transformation, is a mathematical process that maps points from one image onto their corresponding places within another image of the same subject, even though it was taken from a different angle. Points in the first image are precisely transposed to line up with their counterparts in the second image. It is feasible to compute the homography matrix using corresponding points as references given a pair of photos showing the same scene from different perspectives. Any point within one of the images can be effortlessly transferred to its equivalent position within the other image after this matrix has been formed, making it easier to align and compare visual data from various points of view.

The line coordinates in the retrieved frame are then determined using the homography matrix. This is accomplished by multiplying each line's coordinate points in the court's reference image by the matrix, which results in the lines' corresponding coordinates in the extracted frame. To determine the appropriate position in the perspective of the extracted frame, a point in the reference court image represented as a homogeneous  $(x, y)$  coordinate is multiplied by the homography matrix.

The next step is to indicate the court on the extracted frame by drawing these lines at the determined line coordinates in the viewpoint of the extracted frame.

From Figure [3.12](#), the lines representation in the left image should have been obtained.



Figure 3.12: The extracted frame's four points are shown in the left image, while the reference court image and corresponding points are shown in the right image.

However, the cameras used for this Padel court analysis contains fisheye distortion, which makes it difficult to precisely define the court's lines and borders. Fisheye distortion might result in inaccurately drawn lines inside the court because it makes straight lines appear curved. This distortion not only distorts how the court appears visually but also reduces the accuracy needed for line detection and subsequent ball detection algorithms.

### 3.2.3 Ball Position and Trajectory

The location of the ball serves as the foundation for many game-related analysis and insights. A promising and efficient method for ball detection in Padel is KCF object detection algorithm since it is known for its real-time tracking capabilities.

One of KCF's advantages is its effective ability to keep track of an item across multiple frames. KCF should be a great choice for ball tracking because of its ability to adjust to changes in object appearance, scale, and orientation in Padel, where the ball's movement is dynamic and unpredictable. Further enhancing KCF's viability for this application is its resilience in handling partial occlusion, which occurs frequently in Padel when players momentarily block the ball. To precisely map the ball's trajectory and calculate crucial metrics like ball speed, trajectory analysis, and point-winning probabilities, KCF continuously updates the object's position. Furthermore, KCF's real-time capabilities fit with the need for prompt decision-making and analysis in the Padel game.

Despite having promising real-time tracking capabilities, the application of KCF object detection algorithm for Padel ball detection ultimately encountered insurmountable difficulties due to the inherent properties of the Padel ball and the environment in which it operates.

A significant barrier to the success of the KCF-based strategy was the size of the ball and also the resolution of the frames recorded during a Padel match.

Padel balls can seem like mere pixels in the low-resolution frames that are typically accessible in ordinary video recordings of Padel games since they are relatively small. The ability of KCF to track an object depends on the object's ability to be seen clearly and consistently throughout its journey.

In the case of Padel ball identification, KCF found it difficult to reliably identify and track the ball's position precisely due to the scant spatial information offered by the frames. When the ball traveled quickly or when player interactions caused partial occlusions, it was challenging for KCF to maintain robust tracking since the ball frequently appeared as a tiny, intermittent blip within the frames.

The ball's size in relation to the frame resolution continued to negatively impact KCF's performance despite meticulous parameter adjustment and optimization attempts. To escape this problem, a YOLO approach was tried. YOLO has a tennis ball trained class, however the issue remained. The ball appears too small in the frames and with low resolution, making the used approaches to fail.

### 3.2.4 Components Integration

Integrating various components to build a comprehensive AI system for optimizing statistics and player performance insights in Padel involves orchestrating a series of interconnected processes. The key to this AI system's success lies in the seamless integration of these components, which flow is represented in Figure 3.13.

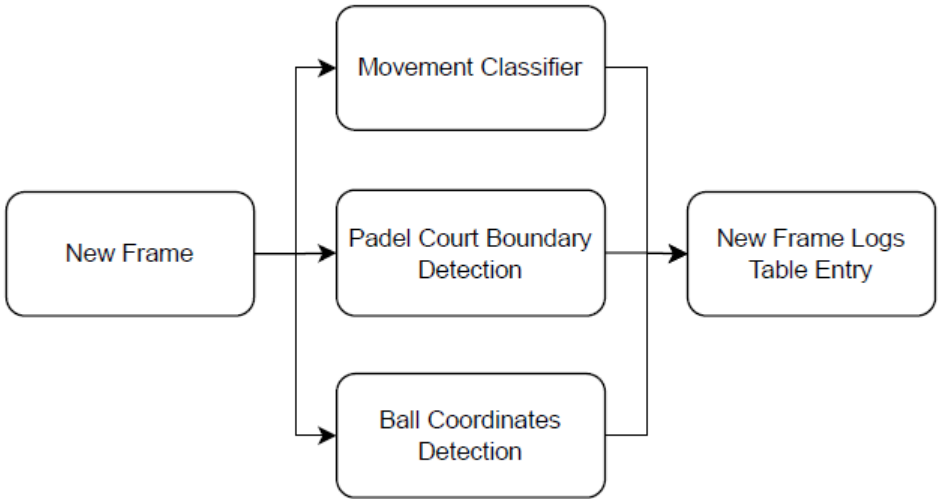


Figure 3.13: AI System's flow representation.

First, movement categorization entails tracking and identifying players' movements and court positions in real time. Ball positioning is established concurrently by monitoring the trajectory, speed, and bounce of the ball. By combining these two elements, it is possible to analyze player tactics and performance in greater detail, gaining insights into shot accuracy, rally dynamics, and player cooperation. Furthermore, contextualizing player motions and ball placement depends on the concurrent computation of the Padel court boundaries. The AI technology makes sure that all player and ball movements stay inside the bounds of the court by precisely recognizing and tracking those boundaries. This data is crucial for determining whether shots are legal, finding errors, and giving a complete picture of games. The simultaneous analysis of these three elements opens the door to a wide range of applications, from automatic game statistics creation to real-time coaching feedback, ultimately improving the entire player experience and advancing the Padel sport.

### 3.2.5 Statistics

Building an AI system for game analysis and statistics production requires creating a complete table that contains data from each frame of the Padel play. The system's foundation is this table, called "frame log", illustrated in the Table [3.1](#), which enables the automatic creation of thorough statistics.

The timestamp at the start of each row in the table shows when the frame was taken. This timestamp serves as a benchmark for following the game's chronological development. The system logs the players' sensed motions for each frame. This could include details like player identification, their locations on the court, and the movements they made (such as running, forehands, and backhands). The ball's exact coordinates within the frame are listed in the table. The camera's perspective and the usual Padel court coordinates are taken into account when recording these coordinates. This enables the charting of ball trajectories, speed estimations, and knowledge of the location of each shot on the court.

The table can have annotations or event markers to draw attention to particular game-related happenings. For instance, timestamps are recorded for occurrences such when a point is scored, a player serves, or a rally is over.

Table 3.1: Example of a failed service executed by player 2 in the frame logs table.

ID	FrameID	Timestamp	PlayerID	Movement	ball_x (cm)	ball_y (cm)	ball_z (cm)	court_left	court_right	court_top	court_bottom
0001	frame_0001	0:00:01	2	Service	600	92	104	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0002	frame_0002	0:00:02	2	Service	579	95	92	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0003	frame_0003	0:00:03	2	Service	585	95	87	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0004	frame_0004	0:00:04	2	Service	599	95	96	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0005	frame_0005	0:00:05	2	Service	600	95	148	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0006	frame_0006	0:00:06	2	Service	605	105	103	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
...	...	...		...	...	...	...	...	...	...	...
0022	frame_0022	0:00:22	null	null	105	85	7	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0023	frame_0023	0:00:23	null	null	102	83	0	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$
0024	frame_0024	0:00:24	null	null	100	79	0	$y = 155x - 37$	$y = -155x + 97$	$y = 20$	$y = 0$

When a player loses a point in Padel, it can be a crucial time in the match. Typically, this happens when the served ball makes contact with the net. In the example of the Table [3.1](#), the exact moment the service was started is recorded in the frame log together with the frame timestamp, ball locations, and player movements. It captures the trajectory of the ball as it moves toward the net and, most importantly, as it hits the net. This occurrence is accurately logged within the context of the Padel court limits thanks to the synchronized time. The moment the system is able to detect that the ball hit the net, it generates statistics regarding the movement that was performed previously. This illustration demonstrates how interesting statistics and insights can be produced automatically from the extensive data in the table. In this example, the AI system can help players and coaches improve their performance in future games by identifying times when service points are lost due to net contact and helping to develop players and strategy. The structured table's ability to support such fine-grained event recording is essential for the pursuit of data-driven advancements in the Padel sport.

## 4 Results Analysis

### 4.1 Movements Classification Model

#### 4.1.1 Metrics

The assessment and testing phase is a crucial stage in verifying the model's performance and its capacity to generalize to new data while training a fine-tuned YOLOv8 model for Padel movement prediction.

To measure the model performance, the following metrics are used:

- Accuracy measures the general correctness of a model's predictions. It is the proportion of cases in the dataset that were successfully predicted to all other instances.
- Recall assesses the model's ability to correctly identify all relevant instances. It is the ratio of true positives (correctly predicted positive instances) to the sum of true positives and false negatives (missed positive instances).
- Precision evaluates the model's ability to minimize false alarms. It is the ratio of true positives to the sum of true positives and false positives (incorrectly predicted positive instances).

Common measures like mAP are also used to measure the model's item detection ability. mAP assesses the accuracy and recall of the model's forecasts at various confidence levels. Greater accuracy in localizing objects and separating true positives from false positives is indicated by a higher mAP score.

- mAP: calculates the accuracy and recall values for each confidence threshold and the average precision (AP) for each class. To get the overall mAP, it then calculates the mean AP for all classes.
- mAP50-95: uses a predefined set of IoU thresholds, typically ranging from 0.5 to 0.95 in steps of 0.05, to determine the Average Precision (AP) for each class independently. The average of these AP scores across all classes is then used to calculate the mAP50-95.

In summary, mAP50-95 focuses on a certain set of IoU thresholds, often incorporating tougher criteria for object detection, whereas mAP gives a thorough evaluation over all confidence levels.

### 4.1.2 Results

In order to gain a clear and concise insight into the model’s classification performance for this classification task, a 3x3 confusion matrix representing two distinct classes (Backhand and Forehand) was constructed, Figure 4.1.

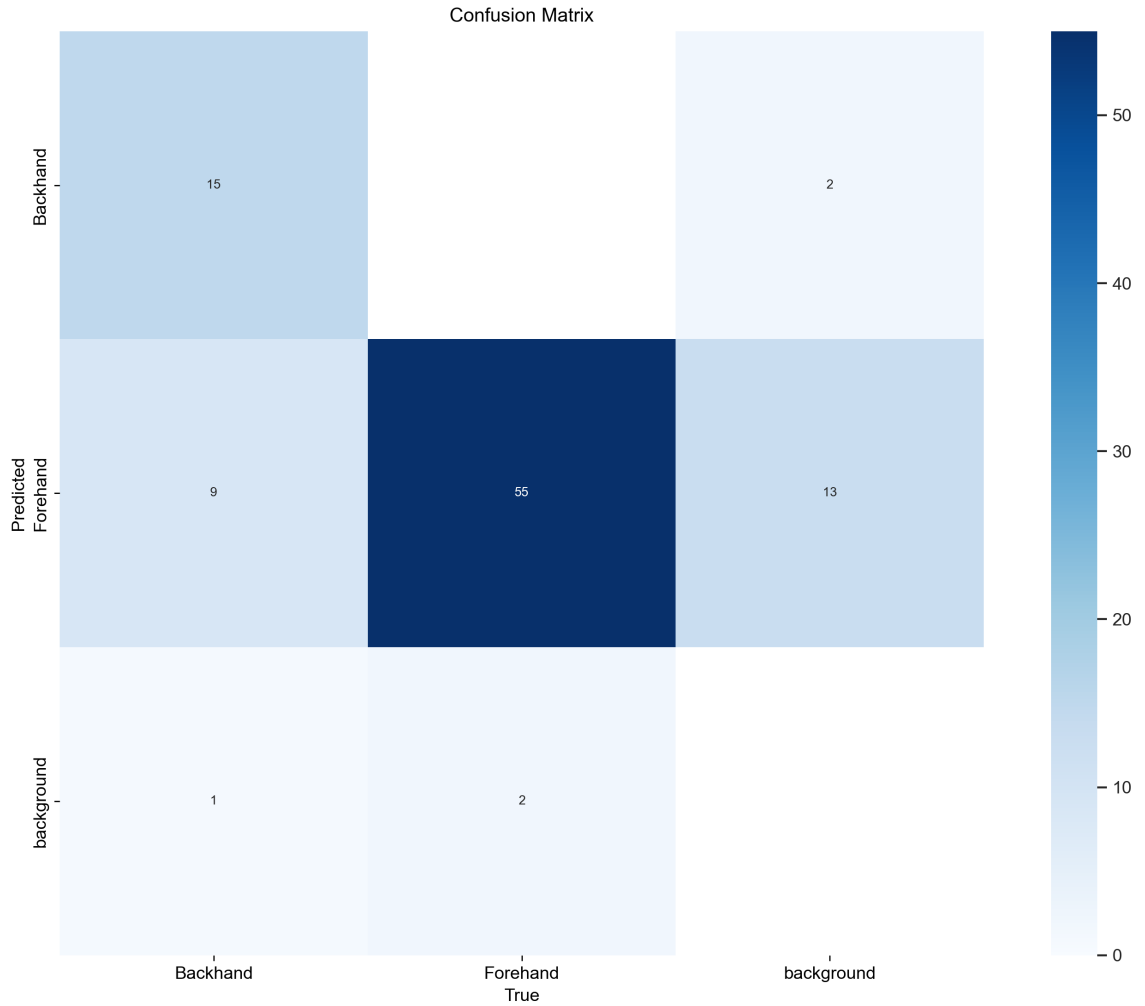


Figure 4.1: YOLO confusion matrix.

The class “Backhand” had 83.3% accuracy, 93.8% recall and 88.2% precision performance whereas “Forehand” had 78.6% accuracy, 96.5% recall and 80.9% precision.

The mAP50-95 was 87.7% for Backhand and 74.4% for Forehand. Overall, the mAP50-95 of the model was 81.1%.



### 4.1.3 Discussion

An interesting observation is that Forehand Class had a higher recall than Backhand Class when also having lower accuracy and precision. Having a higher recall is explained by the fact that the model predicted a high number of true positives. Having lower accuracy and precision is explained by the number of Forehand predictions done by the model when in fact it was a Background. Hence, even though the model had a high number of true positives for Forehand class, it also predicted Forehand when it was nothing in fact a couple of times.

A reason to justify not so consistent metrics is because of the amplitude of the labeled movements that were given as input to the model. The model results show that having a higher amplitude of movements is not good because it might teach the model that potential irrelevant movements might be relevant hits, when it is not true. To fix this, a suggestion would be to remake the labeling reducing the hit movement amplitude up to 3 frames. Also, having a higher amount of labels would be an advantage to improve generalization and consequently model results.

## 4.2 Discussion

### 4.2.1 Research Limitations and Future Works

This master's thesis has explored the use of AI techniques to improve various game elements, from real-time event detection to player performance monitoring, in the dynamic and developing world of AI in Padel. However, there is still a long way to go before AI in Padel is fully utilized. It is critical to recognize as this thesis comes to a close that the topic of AI in Padel offers a broad canvas for further investigation and invention.

The prompt and precise labeling of images used for training and evaluation is a crucial factor. The image identification process must be carried either faster or across fewer frames of the same movement sequence to improve the model's capacity to recognize and categorize Padel movements. This method has the advantage of promoting a higher level of analysis granularity as well as ensuring that the model is trained on temporally coherent and consistent movement patterns. The model gets a more specific understanding of the minute details that distinguish one movement from another by categorizing motions inside a condensed temporal window, ultimately leading to a more accurate and refined movement classifier. This method not only makes the model's training process more efficient, but it also puts it in a position to shine in situations requiring quick and accurate decision-making, which are critical in the fast-paced game of Padel.

For more precise court recognition and subsequent analysis, improving the camera configuration is a key factor. Delineating the court's boundaries precisely is difficult due to the

Fisheye distortion inherent in the current camera setup. Therefore, it is essential to look into methods for either moving the cameras to reduce distortion or applying advanced distortion correction methods. The precision and quality of the input data play a crucial role in how well AI algorithms perform, especially in the areas of object tracking and game analysis.

The detection of the ball inside the frames taken from video recordings was unquestionably one of the major difficulties encountered in this master's thesis. Accurate ball detection proved to be a difficult task given the Padel ball's naturally modest size in relation to the frame's resolution. It is advised that future research projects look into stereo vision systems, in which two cameras are placed carefully in the same plane to take pictures of the Padel court from slightly different angles. A stereo pair of images would result from this configuration, allowing for the estimation of depth information and giving a more thorough perspective of the scene. The restrictions caused by the ball's small size in individual frames could be overcome by using stereo vision to more precisely estimate the ball's position in three-dimensional space. This method has the potential to increase tracking accuracy and ball trajectory prediction in addition to ball detection.

## 5 Conclusion

This research has delved into the world of Padel, a sport that is quickly gaining popularity across the globe, amid the dynamic and changing field of AI applications in sports analytics. The goal to use AI to provide real-time statistics that can improve both player performance and the viewing experience was the driving force behind this work.

To comprehend the complex gameplay that distinguishes Padel, this trip started by investigating the sport. This was done by charting its development across time. The comprehensive examination of the literature then went into the fascinating background and development of AI, the numerous subtypes of AI, the methodologies and algorithms advancing AI, and the crucial role of data in AI applications.

Understanding deep and machine learning techniques, such as a CNN, the YOLO object identification model, transfer learning, position estimation, court detection, and ball detection, laid a crucial basis. The resources needed to address the difficulties of Padel analytics were made available by these technologies.

This study's successful application of movement categorization algorithms was one of its main accomplishments. Results from the analysis of player movement patterns are encouraging. However, the results would be better if the amplitude of the labeled movements were smaller, giving a better understanding of each movement to the movement identification model.

Although our effort to locate and monitor Padel court boundaries has shown promise, it is important to note that this task is still not finished. The intricacies of the Padel court setting, including the camera perspective and fish eye effect, provided difficulties that call for more investigation. This finding of our study emphasizes the necessity for continued research and the possibilities for improving the precision of court border detection with a suggestion of using fish eye distortion techniques in future initiatives.

Despite the best efforts, ball location and trajectory analysis remained a topic under investigation that could not be fully resolved because of the obstacles faced. The ball appearing too small in the captured frames and in some of them appearing with blurred effect did not help with its identification process. These restrictions might be because of camera resolution. A good approach would be using stereo vision and exploring other ways of distributing the cameras along the court. More work needs to be done to give comprehensive real-time statistics, even though this component of our research has significantly increased our understanding of the challenges of tracking swiftly moving objects.

Despite being well planned and in progress, two essential parts, Padel court recognition and ball location and trajectory tracking, remain unfinished. These elements have made it

extremely difficult for our AI system to be fully integrated.

Although these parts are still under construction, the groundwork has been done and improvements have been made. By overcoming these obstacles, this AI system will generate real-time statistics with significantly more accuracy and effectiveness, giving players, coaches, and enthusiasts alike insightful data.

The foundation of this thesis is based on a strong methodology and a distinct future vision, even though the integration of these elements has not yet been fully realized. This is only a key turning point in the ongoing development of AI applications in Padel sports.

## References

- [AllForPadel, 2023] AllForPadel (2023). <https://allforpadel.com/en/blog/10-essential-padel-strokes-n226>, last accessed on 2023-08-26.
- [Almendros, 2009] Almendros, L. J. (2009). Juegos de pala y raqueta en la escuela primaria. *Revista Pedagógica ADAL*, (19):24–29.
- [Almonacid-Cruz, 2011] Almonacid-Cruz, B. (2011). *Perfil de juego en pádel de alto nivel*. [Doctoral Thesis]: Universidad de Jaén.
- [Badiola-Bengoa and Mendez-Zorrilla, 2021] Badiola-Bengoa, A. and Mendez-Zorrilla, A. (2021). A systematic review of the application of camera-based human pose estimation in the field of sport and physical exercise. *Sensors*, 21(18), 5996.
- [Ballard, 1981] Ballard, D. H. (1981). Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122.
- [Bochkovskiy et al., 2020] Bochkovskiy, A., Wang, C.-Y., and Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
- [Bojarski et al., 2016] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. (2016). End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*.
- [Bolme et al., 2010] Bolme, D. S., Beveridge, J. R., Draper, B. A., and Lui, Y. M. (2010). Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE.
- [Boström, 2014] Boström, N. (2014). In *Superintelligence: Paths, dangers, strategies*. Financial Times.
- [Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.
- [Cao et al., 2017] Cao, Z., Simon, T., Wei, S. E., and Sheikh, Y. (2017). Realtime multi-person 2d pose estimation using part affinity fields. *Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7291-7299)*.
- [Castellote, 2012] Castellote, M. (2012). *Atlas ilustrado de pádel*. Susaeta.

- [Ciresan et al., 2011] Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. (2011). Flexible, high performance convolutional neural networks for image classification. In *Twenty-second international joint conference on artificial intelligence*.
- [Devlin et al., 2018] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.
- [Dimitriou, 2021] Dimitriou, I. (2021). Analysis of the symmetric join the shortest orbit queue. *Operations Research Letters*, 49(1):23–29.
- [Donahue et al., 2014] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655. PMLR.
- [Duda and Hart, 1972] Duda, R. O. and Hart, P. E. (1972). Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15.
- [Everingham et al., ] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. <http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html>.
- [Ferreira, 2004] Ferreira, F. (2004). Síntese da história do desporto. *Povos e Culturas*, (9):151–172.
- [Ganin et al., 2016] Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030.
- [Gillmeister, 2008] Gillmeister, H. (2008). Historia del tenis. *ITF Coaching and Sport Science Review*, 15(46):16–21.
- [Gkioxari and Malik, 2015] Gkioxari, G. and Malik, J. (2015). Finding action tubes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 759–768.
- [Gonzalez and Woods, 2008] Gonzalez, R. C. and Woods, R. E. (2008). *Digital image processing*. Pearson Education Ltd.
- [González-Carvajal, 2006] González-Carvajal, C. (2006). *Escuela de Pádel: del aprendizaje a la competición amateur*. Madrid: Tutor.

- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.
- [Haykin, 2009] Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [Henriques et al., 2014] Henriques, J. F., Caseiro, R., Martins, P., and Batista, J. (2014). High-speed tracking with kernelized correlation filters. *IEEE transactions on pattern analysis and machine intelligence*, 37(3):583–596.
- [Hernández Vázquez, 1998] Hernández Vázquez, M. (1998). Deportes de raqueta. *Madrid: Servicio de Publicaciones del Ministerio de Educación y Cultura*.
- [Hough, 1962] Hough, P. V. (1962). Method and means for recognizing complex patterns. US Patent 3,069,654.
- [Hussain, 2023] Hussain, M. (2023). Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection. *Machines*, 11(7):677.
- [in OpenCV using OpenPose, 2018] in OpenCV using OpenPose, M. P. P. E. (2018). <https://learnopencv.com/multi-person-pose-estimation-in-opencv-using-openpose/>, last accessed on 2023-10-03.
- [IPF, 2017] IPF, I. P. F. (2017). *Regulations of the Padel Game*. IPF.
- [James et al., 2013] James, G., Witten, D., Hastie, T., Tibshirani, R., et al. (2013). *An introduction to statistical learning*, volume 112. Springer.
- [Jurafsky and Martin, 2020] Jurafsky, D. and Martin, J. H. (2020). Speech and language processing. 3rd edn. draft. *Online: <https://web.stanford.edu/~jurafsky/slp3>*.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.
- [LearnOpenCV, 2023] LearnOpenCV (2023). <https://learnopencv.com/ultralytics-yolov8/>, last accessed on 2023-12-28.

- [LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.
- [LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [Maas et al., 2013] Maas, A. L., Hannun, A. Y., Ng, A. Y., et al. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, GA.
- [Matas et al., 2000] Matas, J., Galambos, C., and Kittler, J. (2000). Robust detection of lines using the progressive probabilistic hough transform. *Computer vision and image understanding*, 78(1):119–137.
- [McCarthy et al., 2006] McCarthy, J., Minsky, M. L., Rochester, N., and Shannon, C. E. (2006). A proposal for the dartmouth summer research project on artificial intelligence, august 31, 1955. *AI magazine*, 27(4):12–12.
- [Mehmood et al., 2021] Mehmood, K., Ali, A., Jalil, A., Khan, B., Cheema, K. M., Murad, M., and Milyani, A. H. (2021). Efficient online object tracking scheme for challenging scenarios. *Sensors*, 21(24):8481.
- [Murugan, 2017] Murugan, P. (2017). Feed forward and backward run in deep convolution neural network. *arXiv preprint arXiv:1711.03278*.
- [Newell et al., 1972] Newell, A., Simon, H. A., et al. (1972). *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ.
- [Newell et al., 2016] Newell, A., Yang, K., and Deng, J. (2016). Stacked hourglass networks for human pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part VIII 14*, pages 483–499. Springer.
- [NLP, 2019] NLP, T. L. I. (2019). <https://medium.com/modern-nlp/transfer-learning-in-nlp-f5035cc3f62f>, last accessed on 2023-10-02.
- [PadelLab, 2023] PadelLab (2023). <https://www.padellab.co.za/courtconstruction/>, last accessed on 2023-08-26.
- [Pan and Yang, 2009] Pan, S. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.



- [Pearl, 1988] Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan kaufmann.
- [Pfister et al., 2015] Pfister, T., Charles, J., and Zisserman, A. (2015). Flowing convnets for human pose estimation in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 1913–1921.
- [Pineda, 1987] Pineda, F. (1987). Generalization of back propagation to recurrent and higher order neural networks. In *Neural information processing systems*.
- [Rajkomar et al., 2019] Rajkomar, A., Dean, J., and Kohane, I. (2019). Machine learning in medicine. *New England Journal of Medicine*, 380(14):1347–1358.
- [Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788.
- [Redmon and Farhadi, 2017] Redmon, J. and Farhadi, A. (2017). Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271.
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.
- [RookieRoad, 2023] RookieRoad (2023). <https://www.rookieroad.com/padel/how-does-scoring-work-5771030/>, last accessed on 2023-09-25.
- [RoyalNorwich, 2022] RoyalNorwich (2022). <http://www.who.int/mediacentre/factsheets/fs282/fr/>, last accessed on 2023-08-26.
- [Ruder, 2017] Ruder, S. (2017). An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.
- [Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *nature*, 323(6088):533–536.
- [Russell, 2016] Russell, S. (2016). *Artificial Intelligence: A Modern Approach, eBook, Global Edition*. Pearson Education, Limited.
- [Russell, 2010] Russell, S. J. (2010). *Artificial intelligence a modern approach*. Pearson Education, Inc.

- [Sakib et al., 2019] Sakib, S., Ahmed, N., Kabir, A., and Ahmed, H. (2019). *An overview of convolutional neural network: Its architecture and applications*. Preprint.
- [Sánchez-Alcaraz Martínez, 2013] Sánchez-Alcaraz Martínez, B. J. (2013). *Historia del pádel= history of padel*. Materiales para la Historia del Deporte.
- [Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.
- [Sharif Razavian et al., 2014] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.
- [Shazeer, 2020] Shazeer, N. (2020). Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Sonka et al., 2014] Sonka, M., Hlavac, V., and Boyle, R. (2014). *Image processing, analysis, and machine vision*. Cengage Learning.
- [Sutton and Barto, 2018] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.
- [Terven and Cordova-Esparza, 2023] Terven, J. and Cordova-Esparza, D. M. (2023). A comprehensive review of yolo: From yolov1 and beyond.
- [Turing, 1950] Turing, A. M. (1950). Computing machinery and intelligence. *mind*, lix (236), 433–460.
- [Tzeng et al., 2017] Tzeng, E., Hoffman, J., Saenko, K., and Darrell, T. (2017). Adversarial discriminative domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7167–7176.
- [Ultralytics, 2023] Ultralytics (2023). <https://docs.ultralytics.com/models>, last accessed on 2023-12-28.
- [Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

- [Wei et al., 2021] Wei, S., Huang, P., Li, R., Liu, Z., and Zou, Y. (2021). Exploring the application of artificial intelligence in sports training: a case study approach. *Complexity*, 1-8.
- [Wei et al., 2016] Wei, S.-E., Ramakrishna, V., Kanade, T., and Sheikh, Y. (2016). Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732.
- [Wu et al., 2017] Wu, B., Iandola, F., Jin, P. H., and Keutzer, K. (2017). Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 129–137.
- [Wu, 2017] Wu, J. (2017). *Introduction to convolutional neural networks*. National Key Lab for Novel Software Technology.
- [Yosinski et al., 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.
- [Yuen et al., 1990] Yuen, H., Princen, J., Illingworth, J., and Kittler, J. (1990). Comparative study of hough transform methods for circle finding. *Image and vision computing*, 8(1):71–77.