

The College of Wooster

Open Works

Senior Independent Study Theses

2022

Highlights Generation For Tennis Matches Using Computer Vision, Natural Language Processing And Audio Analysis

Alon Liberman

The College of Wooster, alibermanrosenmann22@wooster.edu

Follow this and additional works at: <https://openworks.wooster.edu/independentstudy>



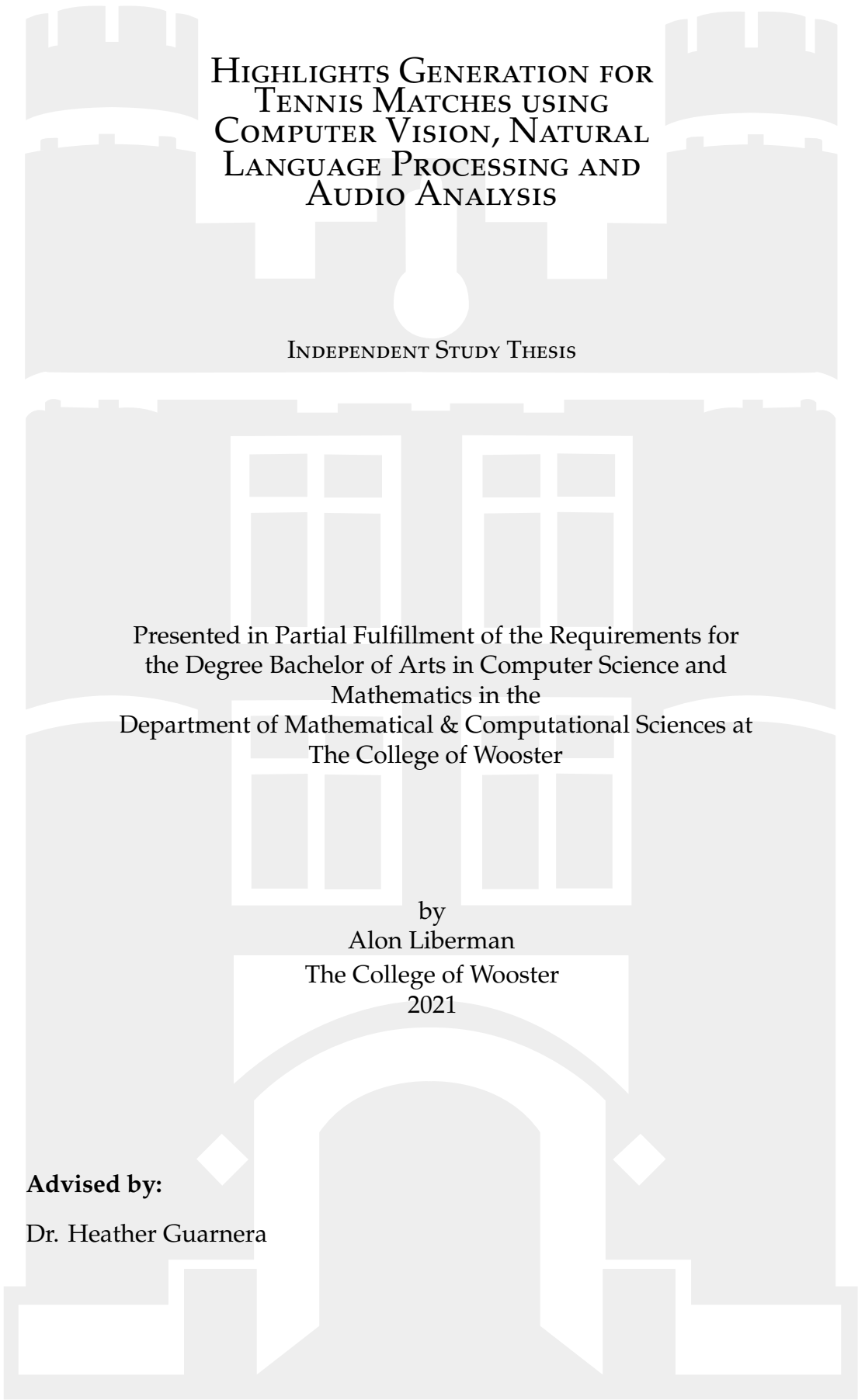
Part of the [Artificial Intelligence and Robotics Commons](#), and the [Other Mathematics Commons](#)

Recommended Citation

Liberman, Alon, "Highlights Generation For Tennis Matches Using Computer Vision, Natural Language Processing And Audio Analysis" (2022). *Senior Independent Study Theses*. Paper 9837.

This Senior Independent Study Thesis Exemplar is brought to you by Open Works, a service of The College of Wooster Libraries. It has been accepted for inclusion in Senior Independent Study Theses by an authorized administrator of Open Works. For more information, please contact openworks@wooster.edu.

© Copyright 2022 Alon Liberman



HIGHLIGHTS GENERATION FOR TENNIS MATCHES USING COMPUTER VISION, NATURAL LANGUAGE PROCESSING AND AUDIO ANALYSIS

INDEPENDENT STUDY THESIS

Presented in Partial Fulfillment of the Requirements for
the Degree Bachelor of Arts in Computer Science and
Mathematics in the
Department of Mathematical & Computational Sciences at
The College of Wooster

by
Alon Liberman
The College of Wooster
2021

Advised by:

Dr. Heather Guarnera



THE COLLEGE OF

WOOSTER

© 2021 by Alon Liberman

ABSTRACT

This project uses computer vision, natural language processing and audio analysis to automatize the highlights generation task for tennis matches. Computer vision techniques such as camera shot detection, hough transform and neural networks are used to extract the time intervals of the points. To detect the best points, three approaches are used. Point length suggests which points correspond to rallies and aces. The audio waves are analyzed to search for the highest audio peaks, which indicate the moments where the crowd cheers the most. Sentiment analysis, a natural language processing technique, is used to look for points where the commentators make positive statements. The software receives a full tennis match with no cuts as the input, and outputs a short summary with the most relevant points. The final software pipeline was tested on three tennis matches from the 2021 US Open for manual validation.

ACKNOWLEDGMENTS

I would like to thank everyone that helped this project become a reality. First, I want to thank my advisor Dr. Heather Guarnera for the guidance throughout this process. This project was developed more efficiently than expected with the access of paid services such as Google Colaboratory Pro, Google Drive One and the Google Speech-To-Text API. Therefore, I want to thank the Copeland Fund for supporting this study. On the other hand, this project relies on countless open source components, which is why I want to thank the open source community as a whole.

I also want to thank my family, for always being there for me. Another special thanks to the tennis team and Phi Sigma Alpha. They are my favourite people from Wooster and the main reason my time here was so invaluable. Specially Ben Foltz, who was always by my side inside and outside the classroom, always present in the most challenging times. Lastly, I want to thank my host family, the McClouds, for making me feel at home since I arrived to Wooster.

CONTENTS

Abstract	v
Acknowledgments	vii
Contents	ix
List of Figures	xi
List of Tables	xiii
List of Listings	xv
CHAPTER	PAGE
1 Introduction	1
1.1 Artificial Intelligence in Sports	2
1.2 Project Purpose	3
1.3 Project Overview	4
2 Background	5
2.1 Computer Vision	5
2.2 Machine Learning	6
2.2.1 Supervised and Unsupervised Learning	6
2.2.2 Image Classification and Object Detection	7
2.3 Deep Learning and Neural Networks	8
2.3.1 Feedforward Neural Networks	9
2.3.2 Recurrent Neural Networks	9
2.3.3 Convolutional Neural Networks	10
2.3.3.1 Convolutional Layer	12
2.3.3.2 Pooling Layer	13
2.3.3.3 Fully Connected Layer	14
2.3.4 Activation Functions	14
2.3.5 Backpropagation & Gradient Descent	16
2.3.5.1 Gradient Descent	16
2.3.5.2 Backpropagation	18
2.4 R-CNN family	19
2.5 Natural Language Processing	22
2.5.1 Sentiment Analysis	23
2.5.2 Speech Recognition	24
2.6 Audio Analysis	25

3	Related Work	27
4	Computer Vision Techniques: Methodology	31
4.1	Shot Detection	31
4.2	YOLO (You Only Look Once)	33
4.2.1	YOLOv3	36
4.3	Tracknet	39
4.4	Hough Transform	45
4.4.1	The Hough Algorithm	46
4.5	CV techniques implementation	47
5	Audio Analysis Techniques: Methodology	51
5.1	Application for improved Point Detection	54
6	Natural Language Processing Techniques: Methodology	57
6.1	Implementation	57
6.1.1	Speech Recognition with Google API	57
6.1.2	Sentiment Analyzer	58
7	Results and Final Software Pipeline	61
7.1	Points Detection Approach and Results	61
7.1.1	Handling False Positives with Audio Analysis	64
7.1.2	Time vs Efficiency: Audio Analysis vs Computer Vision	64
7.2	Final Algorithm and Point Selection	66
7.3	Highlights Generation for Full Tennis Matches	69
7.4	Execution Time	70
7.5	Point Selection	71
7.6	Software Limitations	74
8	Conclusions and Future Work	75
8.1	Threats to Validity	76
8.2	Future Work	76
	References	79

LIST OF FIGURES

Figure	Page
2.1 Simple vs Deep Neural Network	9
2.2 RNN and Feed-Forward Network comparison [3]	10
2.3 Architecture of a CNN [33]	11
2.4 This image shows one filter step. The dot product is taken between the filter and the parts of the input image with respect to the size of the filter [21]	13
2.5 Input and low, mid and high-level features from face recognition task	13
2.6 Example of a 2x2 max pooling operation	14
2.7 Gradient descent to reach local minimum [75]	17
2.8 Forward and backward passes in CNN [16]	18
2.9 R-CNN Architecture [30]	19
2.10 Selection Search. After many iterations, it combines similar regions to make larger regions [70]	20
2.11 Fast R-CNN Architecture [29]	20
2.12 Faster R-CNN Architecture [63]	21
2.13 Region Proposal Network (RPN) arquitetura [63]	21
2.14 Sound waveform [57]	26
2.15 Zoomed in portion of the wave from Figure 2.11 [57]	26
2.16 Low Frequency [34]	26
2.17 High Frequency [34]	26
3.1 Structure of a tennis broadcast [38]	30
4.1 Demonstration of the input image being divided into an $S \times S$ grid [31]	33
4.2 A visual explanation of the IOU metric [52]	35
4.3 YOLO system to produce final detection results [62]. Each cell predicts B bounding boxes and C class probabilities. The highest class probability will determine the class in the class probability map. . . .	35
4.4 YOLOv3 Arquitecture [13]	36
4.5 Illustration of anchor boxes being used to detect two objects in the same grid cell.	37
4.6 YOLOv3 steps. The red cell is responsible for detecting the dog [13].	38
4.7 Output frame showing the tracked ball [39].	40
4.8 Visualization of a zoomed in ball heatmap [39].	41

4.9	VGG16 Architecture [72]	42
4.10	Image showing how unpooling and deconvolution operations are the inverse of pooling and convolution [55].	43
4.11	DeconvNet Architecture [55]	43
4.12	Tracknet Architecture [39]	44
4.13	Image Space and Hough Space. The top images correspond to rectangular coordinates and the bottom images to polar coordinates [45].	46
4.14	Illustration the results of Hough transform on an image. The left picture consists of the input image, the middle is the result of edge detection, and the right shows the lines detected. [42]	47
4.15	Illustration from camera scenes in tennis match and the target points.	48
4.16	Visualization of court, ball and players detection in a single frame from the "Tennis-tracking" project [4].	50
5.1	Graphs of the pressures in two sound waves of different intensities. The more intense sound is produced by a source, the larger the amplitude of the oscillations [56].	53
5.2	Square of the amplitude over time	53
5.3	Peak Detection	54
7.1	Software Pipeline	68

LIST OF TABLES

Table		Page
6.1	Segment of the lists for the Sentiment Analyzer	59
6.2	Sentiment Analyzer input tests	59
7.1	2021 US Open men’s final first set test	63
7.2	2021 US Open women’s final first set test	63
7.3	2021 US Open women’s semifinal first set test	63
7.4	2021 US Open men’s final	69
7.5	2021 US Open women’s final	69
7.6	2021 US Open women’s semifinal	70
7.7	Point Selection for 2021 US Open men’s final	72
7.8	Point Selection for 2021 US Open women’s final	72
7.9	Point Selection for 2021 US Open women’s semifinal	72

LIST OF LISTINGS

Listing	Page
6.1 Sentiment analyzer code snippet	60

CHAPTER 1

INTRODUCTION

Artificial Intelligence (AI) refers to systems or machines that simulate human intelligence processes to perform tasks commonly associated with intelligent beings, such as the ability to reason, understand meaning, generalize, or learn through experience [15]. It is a reality that AI has continuously been replacing manual labor in the last decades. Even though AI did not reach the point of substituting creative thinking, human connection or highly strategic jobs, there is a wide domain where it can surpass humans' capabilities [26].

Computers are less error-prone than humans. They receive a set of instructions and execute them exactly as ordered. Besides, computers can perform repetitive tasks indefinitely whereas humans cannot without suffering from lack of rest, stress, and boredom. While the upfront cost of building and training an AI system can be high, the cost of operation is considerably lower than paying a human to perform the same task. Therefore, AI machines are very profitable in the long run [66].

Businesses and individuals that are able to minimize their dependence on humans can benefit from AI and maximize accuracy and effectiveness at work with negligible costs. One example is the aviation industry, where pilots can now have a helping hand of AI-powered copilots. Another example is the development of chatbots, which are continuously being perfected by tech companies. Nowadays, they are already present in the market, substituting customer service jobs. Chatbots are also

being applied in different areas such as education, healthcare, retail, e-commerce, travel and hospitality [66].

1.1 ARTIFICIAL INTELLIGENCE IN SPORTS

Another area that evolved due to Artificial Intelligence is sports. AI and data analytics have been used to take sports to the next level by improving playability, strategies and audience engagement. AI is leveraged by coaches not only to analyze players' performance to identify their strengths and weaknesses to efficiently maximize their potential, but also to study adversaries by detecting their patterns and develop strategies [48].

Beside performance, another crucial area in sports which was enhanced by AI is health treatment. Being able to diagnose and recover fast from physical and mental issues is extremely important for athletes when they have a demanding schedule with lots of competitions. AI enables coaches to maintain a player's health, fitness, and safety by offering predictive and diagnostic capabilities [24].

In team sports, apart from caring about current players, an essential duty for coaches and managers is to study potential additions to the squad. With Artificial Intelligence, coaches do not need to manually track the stats of recruits and predict future stars anymore. Instead, the staff can monitor and compare complex data in order to estimate a player's market value or gauge whether they would be a good fit for their team [48, 24].

Finally, fan engagement and customer experience has been drastically improved with AI. Nowadays, thanks to Artificial Intelligence, fans have access to apps where they can follow their favorite teams, keep track of their tickets, get notifications for new merchandise drops, locating check-in stations on gamedays, and monitoring the schedule [24].

1.2 PROJECT PURPOSE

The goal of this project is to investigate Artificial Intelligence and apply it to tennis. Tennis is a racket sport that can be played individually against a single opponent or between two teams of two players each. It is one of the sports that has evolved the most through time thanks to the advances in technology. The evolution of the court, racquets and balls over the years led to a drastic improvement in the quality and playability of the game [58, 53].

AI plays a key role in this sport. One of the most famous AI developments that have been incorporated into professional competitions to improve the game is the Hawk-Eye. This is a system for tracking the ball and tracing its path with pinpoint accuracy. Due to its extreme reliability, the most important professional competitions use this technology whenever the umpire is not able to decide whether a ball is in or out. AI is also heavily used for creating complex statistics about tennis matches and analyze metrics such as spin, speed, placement, and the position of players. This provides coaches and fans with very interesting match insights in order to better analyze the match and the players [58, 71, 53].

However, there is still a lot of potential for growth in this field. Many tasks in tennis still rely on manual labor, and will probably be automatized in the future as research in Artificial Intelligence continues to progress. For instance, the production of video summarization. Despite the substantial growth of video data and the demand for tools to accelerate the production of sports highlights, most of the process is still manual and labor-intensive. Wimbledon, for example, is the oldest tennis tournament hosting 250 singles matches over the course of 13 days, producing several hundreds of hours of video [53]. It has not been until 2017 that IBM released their first solution for creating AI-based highlights, and even though some of the biggest tournaments have just started to leverage these solutions, this task is still continuous development [47].

The production of highlights in sports is one of the most essential tasks for broadcast media in order to summarize the exciting moments. Highlights generation is a task which typically requires someone sitting through the entire match and manually selecting the most important moments. Therefore, the focus of this project is to dive into Artificial Intelligence (AI) and develop a highlights generator for official tennis matches in order to automatize this process. Techniques from different areas of Artificial Intelligence are investigated to create a novel approach for gathering the most interesting moments of a tennis match and outputting highlights given a full match with no cuts as the only input.

1.3 PROJECT OVERVIEW

In this paper, a background in topics in Computer Science and Mathematics is given in Chapter 2 in order to introduce the foundations of Computer Vision (CV), Natural Language Processing (NLP) and Audio Analysis (AA). Previous studies that are related to our project will be discussed in Chapter 3. Chapters 4, 5 and 6 present the methodology from CV, AA and NLP respectively, describing the techniques implemented from each field and how they contribute to the project. Chapter 7 focuses on the results of the different techniques and the final software pipeline. Here, three tennis matches from the 2021 US Open are used. Finally, Chapter 8 present the conclusion as well as future work to address the limitations found and enhance our approach.

CHAPTER 2

BACKGROUND

In this section, the foundations of several concepts from Computer Science and Mathematics will be provided to learn the basics of the key areas of this project including computer vision, natural language processing and audio analysis. Techniques from these fields will contribute to our approach for solving the highlights generation task for tennis matches.

2.1 COMPUTER VISION

Computer vision is a subfield of AI that trains computers and systems to interpret and perceive meaningful information from the visual world. This information can be derived from a wide range of sources such as digital photos and footages. Computer vision works with a large amount of image data and analyses them over time until it discerns distinctions and ultimately recognizes some characteristic of the images provided. This field incorporates the skills of deep learning, neural networks and pattern recognition to extract the content of images [40].

Computer vision is a field which is heavily researched and is key for real-world application in areas such as business, entertainment, transportation and healthcare. Thanks to the substantial development of visually instrumented devices like security systems, smartphones and traffic cameras there has been an enormous growth in the flow of visual information which allowed the evolution of this field.

Some of the most established tasks in computer vision are image classification, object detection, object tracking, and content-based image retrieval.

2.2 MACHINE LEARNING

Machine learning is a subfield of artificial intelligence that gives computers the ability to gather data and learn from past experience of the encountered cases, rather than being programmed to perform a certain task. This learning process includes observations of data such as examples, direct experience, or instructions with the aim to look for patterns in it and make better decisions [22].

2.2.1 SUPERVISED AND UNSUPERVISED LEARNING

Machine learning algorithms are often categorized as supervised or unsupervised. A supervised learning algorithm learns from past experience on some task relying on labeled data. A labeled dataset includes both input data and the corresponding expected outputs. After sufficient training, the system is able to provide targets for any new input data. Moreover, the algorithm can compare its outputs with the actual values and compute errors in order to adjust the model accordingly [18].

On the other hand, unsupervised learning deals with unlabeled data with no expected outputs. Even though unsupervised learning does not find the right output, it can explore the data and draw inferences from datasets to describe hidden patterns within datasets.

The difference between these two types of learning leads to very different use cases. For example, supervised learning is appropriate for classification problems where the data needs to be assigned to different categories such as detecting different types of fruits or classifying emails as spam or not. Classification algorithms include linear classifiers, support vector machines and decision trees. Another very common

technique for supervised learning is regression. Regression models are helpful to understand the relationship between dependent and independent variables, in order to predict numerical values based on different data points.

Unsupervised learning are used for three main tasks: clustering, association and dimensionality reduction. Clustering is a technique used to group data based on similarities and differences, and has applications in market segmentation and image compression. Association uses specific rules to find relationships between variables in a dataset. This method is key for basket analysis and recommendation engines. Finally, dimensionality reduction is used to reduce the number of features in a dataset such that the amount of data is manageable while still preserving the data integrity. This technique can be used to remove noise to improve picture quality.

2.2.2 IMAGE CLASSIFICATION AND OBJECT DETECTION

Image classification and object detection are among the most important tasks in Computer Vision. Image classification involves predicting the class of one object in an image. For instance, an image classification algorithm can be trained with a labeled dataset composed of images of apples and bananas in order to classify new input images. Image classification is a supervised learning problem since the dataset needs to be labeled to train the classifier [9].

Object localization refers to the task of identifying the location of the object in an image and drawing bounding boxes around them. Object detection combines localization and classification, locating the presence of objects with both a bounding box and a class label. One further extension to these computer vision tasks is object segmentations, where the objects recognized are highlighted by specific pixels of the objects to fit their shape. These techniques has been used in many applications, including human face recognition, retail checkout recognition, automated driving, and automatic monitoring systems [52].

2.3 DEEP LEARNING AND NEURAL NETWORKS

Deep learning (DL) and artificial neural networks (ANN) are both subfields of artificial intelligence. Deep learning is also currently considered a subset of machine learning. Here, instead of teaching computers to process and learn from data, with deep learning the computer trains itself to do so.

DL and ANNs are directly related to each other, where artificial neural networks are the backbone of deep learning. ANNs can be defined as a computational nonlinear model based on the neural structure of the brain that is able to learn to perform tasks such as classification or prediction. They are organized in three interconnected layers that send information to each other: input, hidden and output. These layers are connected via nodes or "neurons", creating the form of a network. Every node has its own small sphere of knowledge that applies to the input information, and the neuron in the output layer contains the final output. The output layer can also have more than one neuron. For instance, in multi-class classification each output node represents the probability for a specific class. A neural network is fed huge amounts of data on the topic it will be used for, providing both the inputs and the expected outputs [2].

Previously, models consisted of the simplest form of neural networks with just an input layer, one hidden layer and an output layer. It was not until computers could process neural networks with multiple hidden layers that we could talk about "Deep Neural Networks" or deep learning. The difference between simple and deep NNs is depicted by Figure 2.1.

There are many types of neural networks used in deep learning, each with their unique strengths and use cases. Therefore, it is very important to have a foundation of the most important ones in order to choose the appropriate network for a specific problem. Neural networks are utilized by a wide variety of industries, and some notable applications include speech-to-text transcription, handwriting

recognition, weather prediction, facial recognition, chatbots, stock market prediction and delivery route planning and optimization [11].

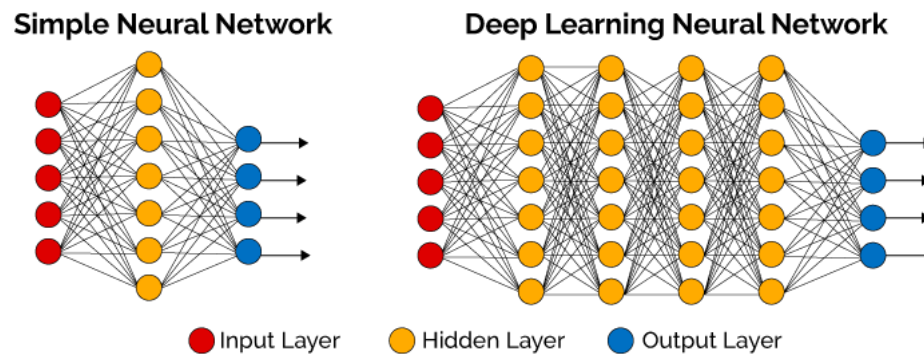


Figure 2.1: Simple vs Deep Neural Network

2.3.1 FEEDFORWARD NEURAL NETWORKS

The feedforward neural network (FNN) is one of the simplest neural networks. Here, the data moves only in one direction from the first layer until the output. Every neuron in a layer is connected with all the neurons of the previous layer, and each connection has a specific weight. In a FNN, the sum of the products of the inputs and their weights are calculated and fed to the output. One important characteristic of this type of network is the activation function. This is a function attached to each neuron as a “gate” and basically determines whether it should be activated or not, based on whether its input is relevant for the prediction or not. Activation functions introduced nonlinear properties to the network, which help it learn more complex data and learn almost every function [2].

2.3.2 RECURRENT NEURAL NETWORKS

A very powerful type of neural networks are recurrent neural network (RNN). The key characteristic of these networks is that the connections between neurons make a directed cycle. From each step to the next, each node retains information that it

had in the previous step in a way that nodes can be viewed as memory cells while doing computations. In Figure 2.2 it is possible to see a RNN that has a recurrent connection on the hidden state. This looping constraint ensures that the output of a particular layer is saved and fed back to the input such that sequential information can captured in the input data. This property makes RNNs suitable for certain data types such as time-series, text, and biological data that contain sequential dependencies among the attributes [2].

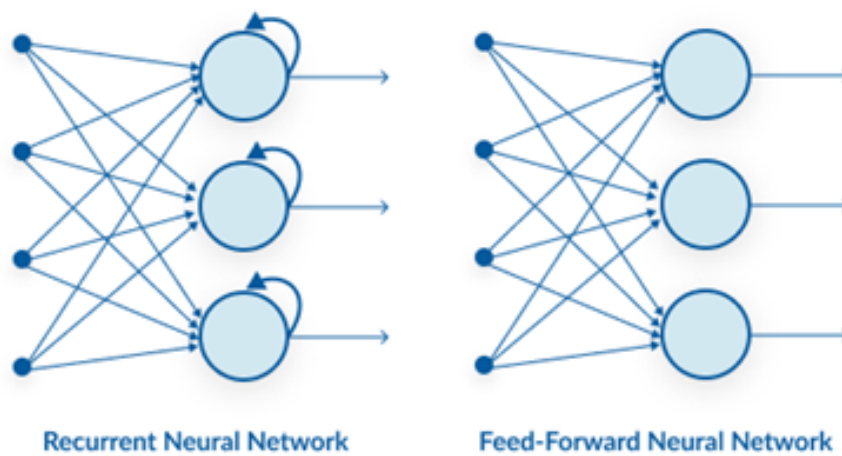


Figure 2.2: RNN and Feed-Forward Network comparison [3]

2.3.3 CONVOLUTIONAL NEURAL NETWORKS

Convolutional neural networks (CNN) are the most relevant type of NNs for this project. CNNs are very powerful networks that are suitable to work with grid-structure inputs with strong spatial dependencies in different regions of the grid. This type of network excels is with 2-dimensional images, where adjacent spatial locations contain patterns that can be studied to extract information from images.

An image is understood by computers as sets of pixels, each represented by one or more color components with intensity values denominated channels. Grayscale images are single channel images, as the only colors are shades of gray. Secondly,

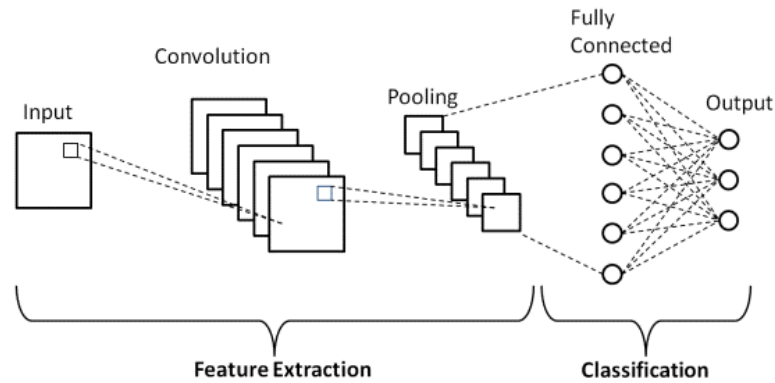


Figure 2.3: Architecture of a CNN [33]

RGB (Red Green Blue) is an additive color model which is very popular in web design. CMYK (Cyan Magenta Yellow Black) is a subtractive color model composed of four channels, which is typically used for commercial printing and painting.

A convolutional neural network is a type of feed-forward neural network that has a similar structure to the networks describe above, but differs from it as its hidden layers referred to as “convolutional layers”, make the network capable of detecting patterns in different regions of the data. For instance, when working with images, there are a lot of patterns in it such as multiple edges, shapes, textures and objects which can be detected by different filters or kernels present in the layers that have learnable weights and biases. Each filter takes some inputs, performs convolution, and optionally follows it with a nonlinear computation [17]. The term ‘convolution’ in CNN denotes the mathematical function of convolution which is a special kind of linear operation wherein two functions are multiplied to produce a third function which expresses how the shape of one function is modified by the other [33]. Figure 2.3 shows a typical CNN architecture, which is described below.

A convolutional neural network is composed by the following layers:

2.3.3.1 CONVOLUTIONAL LAYER

The convolutional layer is the core building block of a CNN which does most of the computational work (convolution) to extract the features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter or kernel of a particular size $n \times n$. These filters typically have small sizes such as 3×3 or 5×5 to detect features. Figure 2.4 shows a single filter step. By sliding the filter over the input image from left to right by a specific stride, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($n \times n$). After the filter slides through the entire width of the image matrix, it moves back to the left and down by the stride value. This produces a feature map which gives us information about the image such as the corners and edges and this feature map is fed as input to other convolutional layers to learn other features of the input image [33, 21].

Figure 2.5 shows the features extracted when doing face recognition after many convolutional layers. The deeper the CNN layer goes, the more complex patterns are extracted. First, edges are recognized, then parts of the face, and finally complete faces.

Padding is typically used when the filters do not cover the entire data. In this case, all elements that fall outside the input matrix are set to zero. For instance, a padding of 1 adds a column or row to every side. The convolution can output either an equally sized matrix or a smaller one depending on the stride and padding values [21, 52]. The following formula calculates the dimension of the output matrix:

$$\left\lfloor \frac{n_a - n_k + 2p}{s} \right\rfloor + 1 \quad (2.1)$$

where $n_k \times n_k$ is the size of the filter, $n_a \times n_a$ corresponds to the input matrix, s to the stride and p the padding.

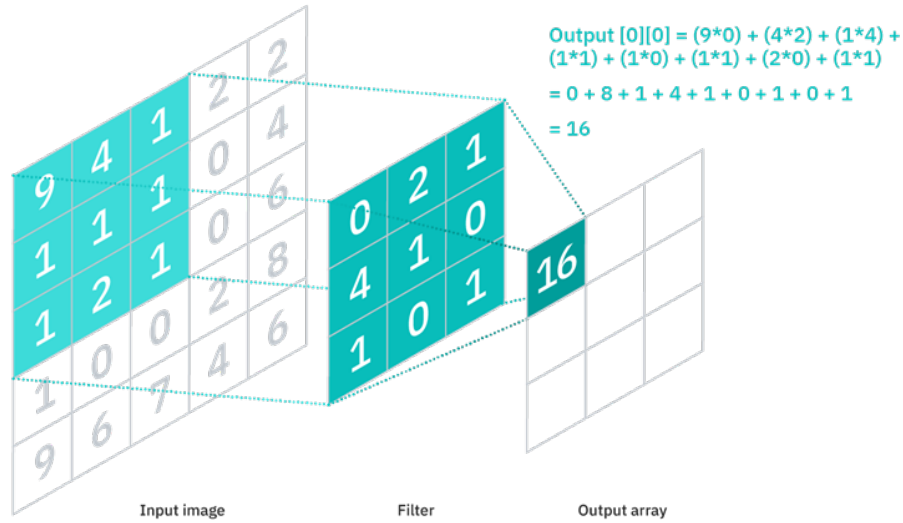


Figure 2.4: This image shows one filter step. The dot product is taken between the filter and the parts of the input image with respect to the size of the filter [21]

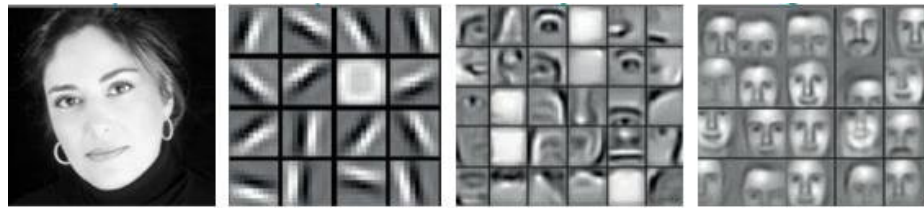


Figure 2.5: Input and low, mid and high-level features from face recognition task

2.3.3.2 POOLING LAYER

The pooling layer reduces the dimensionality of the convolved feature map to reduce the computational costs without losing the most important information. The input images are divided into a set of nonoverlapping rectangles, and each region is down sampled by a non-linear operation such as average and maximum. This layer achieves better generalization, faster convergence, limit risk of overfitting, is robust to translation and distortion and is usually placed between convolutional layers [17].

Max and average are the two main types of pooling. Max pooling consists of selecting the maximum values from every region and send it to the output array as the filter moves across the input [21]. On the other hand, average pooling calculates

the average value within the receptive field and send it to the output array. Figure 2.6 shows a simple example of max pooling on a 4×4 matrix using a 2×2 filter and a stride of 2.

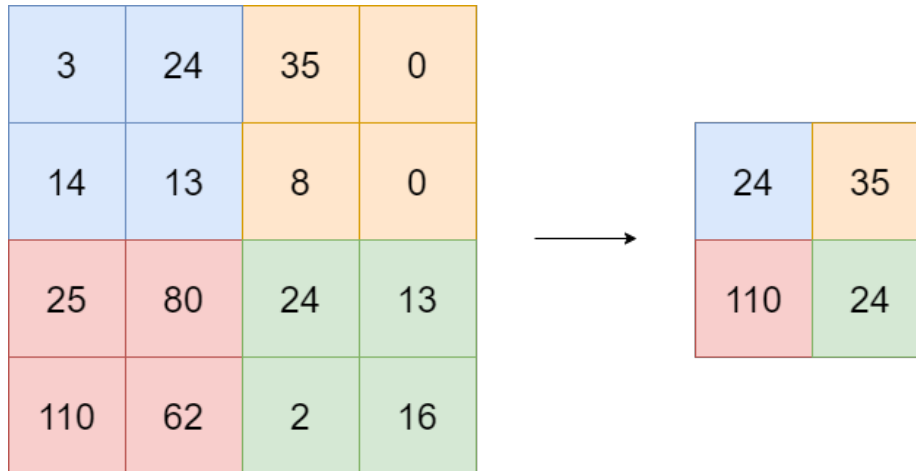


Figure 2.6: Example of a 2x2 max pooling operation

2.3.3.3 FULLY CONNECTED LAYER

CNNs detect lower-level features which are used by the convolutional layers to detect the higher-level features. Then, the fully connected layers connect every neuron in one layer to every neuron in another layer with the respective weights and biases. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture. Here, the layer performs the task of classification based on the features extracted [33].

2.3.4 ACTIVATION FUNCTIONS

Layers usually leverage activation function in order to define how the weighted sum of the input is transformed into an output from the nodes. There are several commonly used activation functions such as the ReLU, softmax, \tanh and the sigmoid functions [33].

The output layer activation function is chosen based on the type of prediction that needs to be solved. Sigmoid activation is appropriate for binary classification, where there is only one node in the output layer. It is also appropriate for multilabel classification, where there are two or more mutually inclusive classes and the output layer contains one node per class. Softmax activation is useful in multiclass classification, where different classes are mutually exclusive [49].

A neural network also contains activation functions in its hidden layers. Nowadays, ReLU is the general activation function used in most cases for the hidden layers of Neural Networks. Some neural networks such as RNNs still use \tanh and sigmoid activation functions. In CNNs, the convolutional and pooling layers tend to use ReLU functions, whereas fully connected layers usually use a softmax activation function to classify the inputs producing probabilities from 0 to 1 [49].

The ReLU function is defined as:

$$f(x) = \max(0.0, x) \quad (2.2)$$

If the input x value is negative, the returned value will be 0.0. Otherwise, the function returns x .

The sigmoid function takes any real value and maps it to a decimal in the range $[0.0, 1.0]$. Therefore, the bigger the input, the closer the output will be to 1.0. The formula of this activation function is

$$f(x) = \frac{1.0}{1.0 + e^{-x}} \quad (2.3)$$

The \tanh function is similar to sigmoid, with the difference that the range of the outputs is in the range $[-1.0, 1.0]$. This function can be calculated as

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \quad (2.4)$$

Lastly, the Softmax activation function is a multi-class classifier takes an input vector of real values x_0, \dots, x_n and outputs a vector of the same size with elements in the range $[0.0, 1.0]$ that sum to 1 [74].

$$f(x) = \frac{e^{x_i}}{\sum_{j=1}^n e^{x_j}} \quad (2.5)$$

The bottom term of the equation applies normalization in order to have all the output values of the function sum to 1 to have a valid probability distribution.

2.3.5 BACKPROPAGATION & GRADIENT DESCENT

Backpropagation is a key algorithm for neural networks. It uses gradient descent in order to tweak the weights of the model while training. This technique consists of going backwards through the neural network to find the partial derivatives of the error with respect to the weights. These derivatives are then used by gradient descent to adjust the weights up or down in order to decrease the error.

2.3.5.1 GRADIENT DESCENT

In vector calculus, a gradient [28] is the vector field λf at a point p and its components are the partial derivatives of f at a point p . This vector indicates the direction with the largest changing value at p and is defined as:

$$\text{grad} f = \Delta f(x, y, z) = \left\langle \frac{\partial f}{\partial x}(x, y, z), \frac{\partial f}{\partial y}(x, y, z), \frac{\partial f}{\partial z}(x, y, z) \right\rangle \quad (2.6)$$

where the components correspond to the partial derivatives with respect to the variables x , y and z .

Gradient descent (or steepest descent) is a first-order iterative optimization algorithm used to find a local minimum of a differentiable function. This algorithm consists of stepping in the opposite direction of the gradient at the current point,

which corresponds to the direction of the steepest descent. On the other hand, taking steps in the same direction of the gradient can be done to achieve a local maximum. The gradient descent method is used to define the distinction between two values to train neural networks [28]. An illustration of gradient descent is showcased in Figure 2.7.

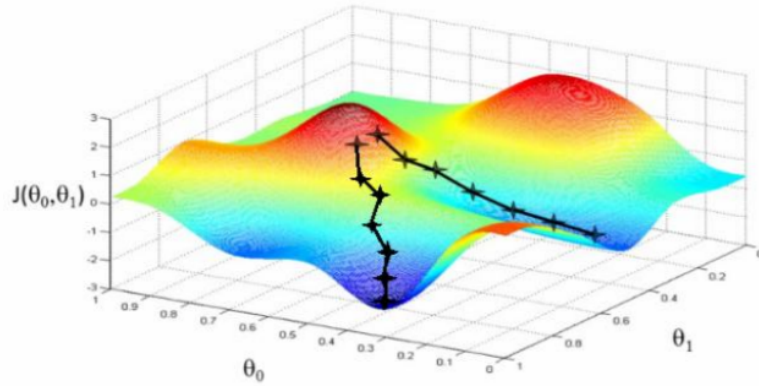


Figure 2.7: Gradient descent to reach local minimum [75]

The algorithm is based in the following process [75]:

1. Pick an arbitrary starting point.
2. Compute the gradient
3. Move in the opposite direction of the gradient by the step size times the gradient at the current point.

The general formula [1] for obtaining p_{n+1} from position p_n is :

$$p_{n+1} = p_n - \alpha \Delta f(p_n) \quad (2.7)$$

where α is the step size.

It must be noted that the step size must be carefully selected. If it is too high, the algorithm will overshoot the minimum and keep bouncing without reaching it. If it is too small, a lot more steps will be necessary to reach the minimum [28].

2.3.5.2 BACKPROPAGATION

As Figure 2.8 suggests, a forward pass of a CNN receives inputs x and y and applies $f(x, y)$ to output z . Backpropagation refers to the backward pass in a convolutional layer. The idea is to compare the output of the forward pass with the actual value, update the parameters (weights, biases and filters) using the gradient descent technique and repeat the forward propagation with the new parameters [8].

Here, the gradient of the loss function $\frac{\partial L}{\partial z}$ respect to z (output gradient) is propagated backwards and multiplied by the local gradients (chain rule) $\frac{\partial z}{\partial x}$ and $\frac{\partial z}{\partial y}$ in order to calculate the gradients $\frac{\partial L}{\partial x}$ and $\frac{\partial L}{\partial y}$ of the inputs x and y (input gradient).

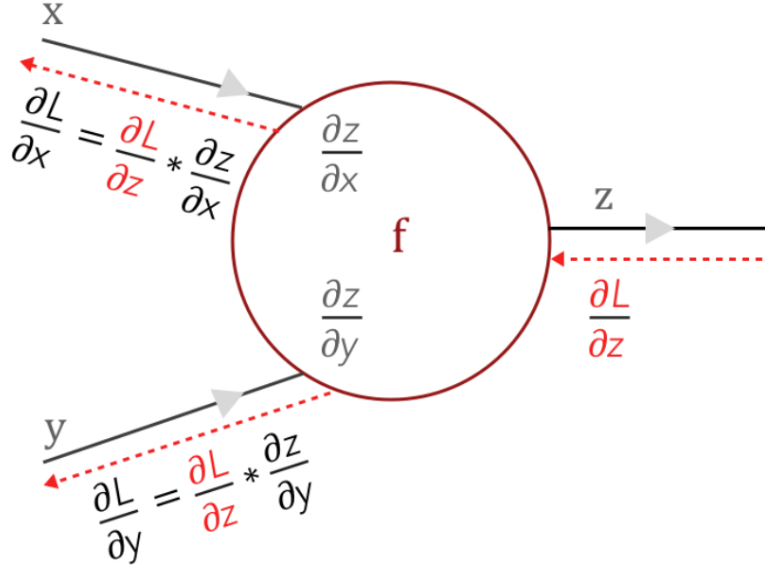


Figure 2.8: Forward and backward passes in CNN [16]

The obtained input gradients are then going to be used to calculate the gradient of the parameters. For instance, equation 2.8 shows how to obtain the weights gradient. The gradients of the parameters are going to indicate in what direction and by how much to update them [75].

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} * \frac{\partial y}{\partial w} \quad (2.8)$$

Finally, the parameters can be updated using equation of gradient descent [68]:

$$newParameter = oldParameter - (learningRate * gradientOfParameter) \quad (2.9)$$

2.4 R-CNN FAMILY

The traditional object detection era ended with the rise of deep learning and the introduction of the multi-stage R-CNN (regions with convolutional neural networks) family. R-CNN models look at parts of the image which are likely to contain objects. The first model was R-CNN, then Fast R-CNN improved certain aspects, and thirdly, Faster R-CNN became the state-of-art multi-stage object detector. This last model still remains one of the most accurate object detection models [30, 29, 63].

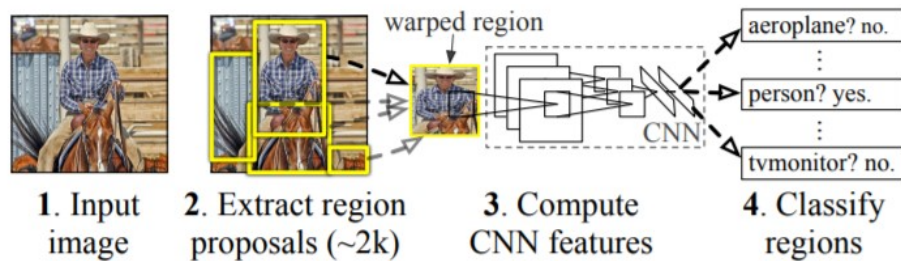


Figure 2.9: R-CNN Architecture [30]

Figure 2.9 shows the architecture R-CNN, which combines different algorithms. First, it uses a selection search algorithm to choose 2000 regions of interest (RoI) which could contain objects using patterns such as varying scales, colors, textures and enclosure. Figure 2.10 shows the resulting regions for an image after many iterations. Secondly, each of these regions is processed through a CNN to obtain feature maps. Lastly, it performs classification to obtain object classes and regression to obtain bounding boxes. R-CNN was a novel method at the time, even though it is considerably slow [30].

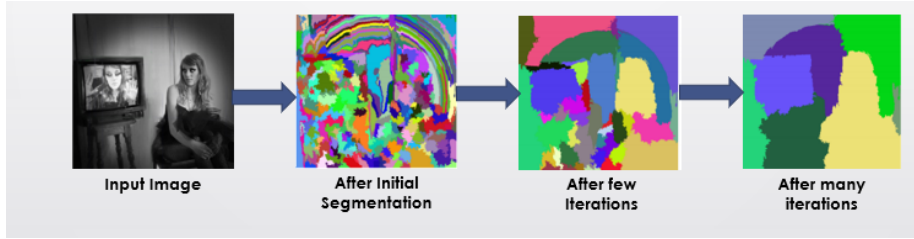


Figure 2.10: Selection Search. After many iterations, it combines similar regions to make larger regions [70]

Fast R-CNN was the next iteration of R-CNN, and it brought a few improvements. Here, instead of feeding each of the 2000 regions to the CNN, the entire image is passed once. Therefore, only one combined feature map is obtained, and the region proposals are identified from it using selection search. Then, a RoI pooling layer is used to reshape the regions of interest to a fixed size. This type of pooling layer performs max pooling on inputs with non-uniform sizes to produce a smaller output with a fixed size. Finally, they are fed to a fully connected layer with two branches: a softmax classifier to obtain object classes and a bounding box regressor for the box coordinates. The diagram for this model [29] is depicted in Figure 2.11.

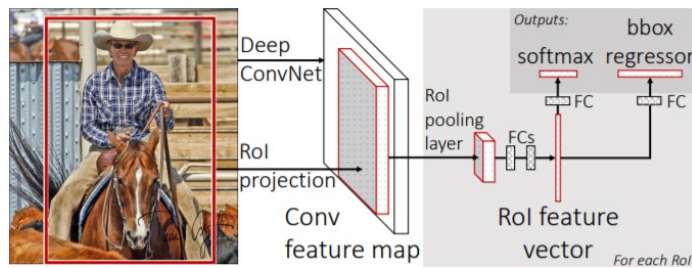


Figure 2.11: Fast R-CNN Architecture [29]

The main problem that remained with Fast R-CNN is the time consuming step of selective search for the RoI. The main improvement in Faster R-CNN was to replace the selection search algorithm with a convolutional network called region proposal network (RPN). RPN takes image feature maps as input and outputs a set of RoI using a sliding window approach. Figure 2.13 shows the sliding window at a spatial location of the feature map. At each spatial window, there are many

predetermined boundary boxes or "anchor boxes" of different scales and aspect ratios in order to detect objects of different shapes and sizes. Then, the network feeds into fully connected layers to predict the object probability and the bounding box for each anchor box. This improved design substantially increased the speed, making it useful for real-time object detection [63]. Figure 2.12 showcases the Faster R-CNN model.

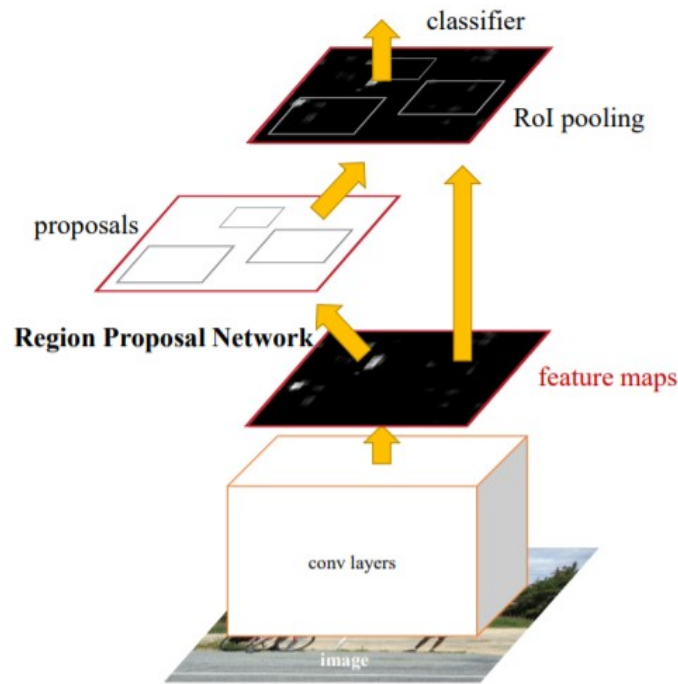


Figure 2.12: Faster R-CNN Architecture [63]

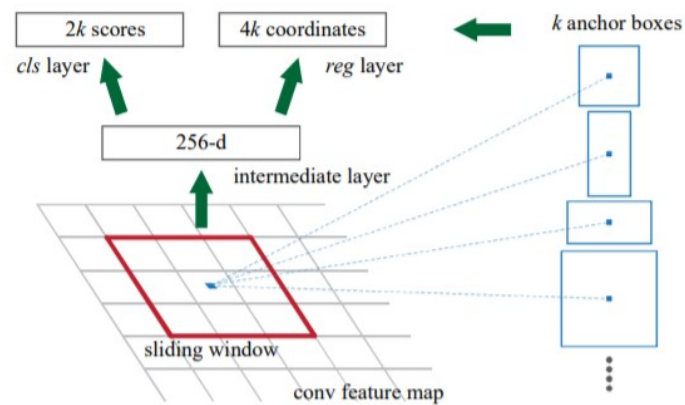


Figure 2.13: Region Proposal Network (RPN) architecture [63]

2.5 NATURAL LANGUAGE PROCESSING

Natural language processing (NLP) is an intersection of computer science, artificial intelligence and linguistics. The goal of this field is to give computers the ability to read, understand and derive meaning from natural languages. Typical applications in NLP include speech recognition, spoken language understanding, lexical analysis, text generation, information retrieval, among others [46].

The first wave of NLP started back in the 1950s, and the approaches were based on the belief that knowledge of language in the human mind is fixed in advance by generic inheritance. These were called rationalist approaches and consisted of designing sets of handwritten rules to incorporate knowledge and reasoning mechanisms into intelligent NLP systems. Here, the premise is that computers emulate natural language understanding by applying those rules to the data it is confronted with.

The second wave started in the 1990s. This wave was characterized of empirical approaches that assume the human mind only begins with general operations for association, pattern recognition and generalization. Here, sensory input was necessary to learn the structure of natural language. Some of the generative models include the hidden markov model, the IBM translation models and the head-driven parsing models to learn the regularities of natural languages from large amounts of data. A few years later, representative discriminative models such as the perceptron, supporting vector machines, the maximum entropy model and minimum classification error became widely used in NLP.

This empiricism in Artificial Intelligence was based on data-intensive machine learning. However, it is now classified as shallow due to the lack of abstractions provided by multiple layers of data. Although the shallow discriminative models were a big improvement from the first wave, they were far from human-level performance and were not good enough. This was mainly due to the inability of

those models to absorb sufficient amounts of training data. The biggest revolution in NLP occurred in the third wave in the 2000s when deep neural network-style machine learning methods became widespread.

2.5.1 SENTIMENT ANALYSIS

Sentiment analysis is one of the key application areas of NLP. It is a text classification problem with the objective of developing algorithms to determine the text's positive, neutral, or negative connotation. However, it can focus on feelings, urgency, or intentions. There is a wide variety of applications where sentiment analysis is highly effective including analyzing reviews, detecting trends, conducting market research, and posts [73].

There are three approaches for sentiment analysis algorithms. Rule-based systems are based in sets of human-crafted rules. These rules include NLP techniques such as stemming, tokenization, part-of-speech tagging, parsing and lexicons. A basic rule-based system would consist on defining two lists of polarized words with their respective scores, and then calculating the sentiment score based on the count of positive and negative words in the input text. The aggregate score would be positive if the sum of positive scores is greater than the negative, and vice versa.

Automatic approaches rely on machine learning techniques. Sentiment analysis is typically a classification problem. Therefore, it requires a labeled dataset for the training process in order to learn the association between a particular input and the respective output. Feature extraction is the first step in a machine learning text classifier. Traditional approaches consist of bag-of-words or ngrams with frequency. However, newer approaches are based on words embeddings achieved better performance by allowing similar words to have similar representations. After extracting the features, the next step for automatic approaches is applying a

classification algorithm. This is typically achieved with statistical models such as naïve bayes, logistic regression, support vector machines or neural networks.

Although rule-based algorithms are simple to implement, they often overlook the complexities of text and word combinations. The choice between rule-based and automatic algorithms is made depending on how advanced the sentiment analysis model needs to be. The third approach is a hybrid system, and consists of merging rule-based and automatic techniques. Although these are generally more complex to build, the outcome is that they perform more accurately than using either a rule-based or an automatic approach.

Sentiment analysis is a hard problem due to all the challenges it faces. Some of the most common difficulties include handling sarcasm, context, subjectivity, tone, comparisons, emojis and idioms. Moreover, humans can also struggle in order to label the data properly [44].

2.5.2 SPEECH RECOGNITION

NLP techniques can also be applied to audio data. The goal of speech recognition is to transcribe oral data into a stream of words. Nowadays, many industries benefit from speech technology in their systems. For instance, mobile devices companies increasingly develop virtual agents to perform voice search tasks. Apple's Siri, Google Assistant, Microsoft's Cortana and Amazon's Alexa are some of the most remarkable examples [23].

One of the traditional models for speech recognition is N-grams, which consists of assigning probabilities to phrases and use the previous n number of words as context to get the next word. Another approach are Hidden Markov Models (HMM), which works in the opposite direction as N-gram models. It uses probability and statistics to look forward and predict the word. Nowadays, these models have been replaced by algorithms that take advantage of deep learning and neural networks.

These models are trained with a large amount of audio data and the expected stream of words written by a human transcriptionist [65].

2.6 AUDIO ANALYSIS

Audio analysis is concerned with extracting information from audio signals captured by digital devices and is applied in many areas such as healthcare, enterprise, and smart cities. Some of these applications are costumer satisfaction, content analysis, medical diagnostic aids and patient monitoring [20].

Audio analysis can leverage machine learning for a variety of tasks, including recognizing between different types of sounds, segmenting an audio signal to homogeneous parts or group sound files on their content similarity. Here, the objective is to go from the plain audio data to a higher-level representation to extract features that provide useful information such as different speakers, events, emotions and musical genres [20].

Sound is typically represented as waves in most types of sound recording software. Sound waves have four physical properties. The amplitude is the height of the sound wave and is measured in decibels (dB). The waveform representation shows how the amplitude of the sound changes over time. The amplitude usually oscillates above and below the x axis, as it can be seen in Figure 2.14 . Figure 2.15 shows a zoomed in portion of the previous audio wave. The wavelength is the horizontal length of one wave cycle. It is the distance from the crest of one wave to the next nearest crest. The propagation velocity is the speed at which the sound propagates through different materials, and thus is dependent upon the medium's characteristics. For instance, the propagation velocity of average human tissue is 1540 m/sec [34].

The total number of cycles produced in one second is called the frequency of the

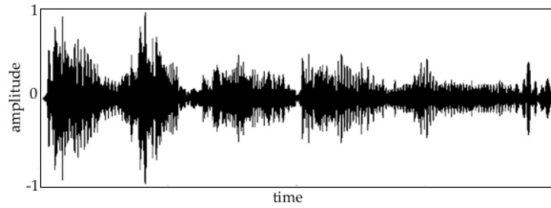


Figure 2.14: Sound waveform [57]

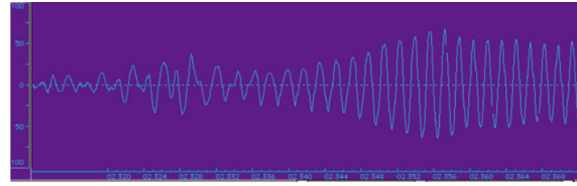


Figure 2.15: Zoomed in portion of the wave from Figure 2.11 [57]

wave, and it is measured with the Hertz (Hz) unit. Frequency can be calculated by dividing the wavelength by time. Humans' hearing range goes from 20 to 20,000 Hz. Sound waves with frequencies greater than 20,000 Hz are denominated ultrasound. Figures 2.16 and 2.17 show the difference between low and high frequencies.

Sound descriptors provide useful information about sound files. A sound descriptor is a numerical description of an aspect of the sound. They can describe the entire sound (global) or variations within it (time-varying).

The simplest descriptor is duration and is a global descriptor. Sound energy is another global descriptor and suggests the total sound energy in the sound file. It can be calculated by computing the root-mean-square value of the waveform. It is also possible to calculate the energy in different sections of the sound file, acting as a time-varying descriptor.

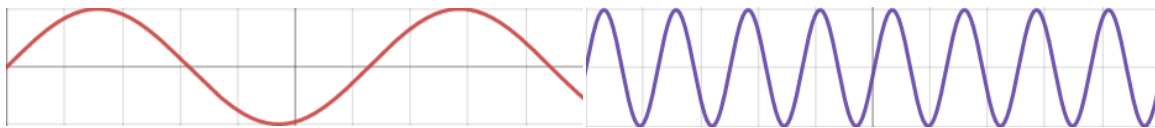


Figure 2.16: Low Frequency [34]

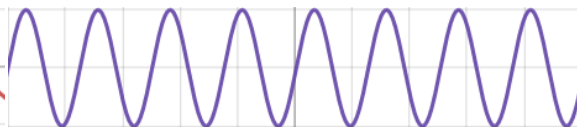


Figure 2.17: High Frequency [34]

CHAPTER 3

RELATED WORK

Video summarization is a field that has a long history of research. The objective of video summarization is to generate short videos that summarize the most important content of full-length videos. One area of video summarization is automatic trailer generation, which involves detecting the most relevant scenes in a movie. Here, the content with the highest emotional impact is retrieved based on the genre, such as the most suspenseful scenes in horror movies [51].

In this project, our focus is on highlights generation for sports videos. Sports highlight detection is a field in continuous research. Previous studies implemented sports video summarization with different approaches such as using audio cues, visual cues and audio-textual cues [51, 5, 38].

Audio cues were leveraged to create highlights in multiple sports [51, 5, 38]. However, the audio contents of each sport can have different characteristics. For instance, in tennis matches the points are usually played in complete silence with the crowd cheering after they finish, whereas sports like soccer and rugby have continuous applause and whistles in the entire games. Moreover, the sounds of the ball, dribbling and players are also typically unique to each sport.

Anant Baijal et al. presented an approach [5] for a highlights generator applied for rugby games using audio information. In this study, multi-stage classification was developed for detecting key acoustic events that suggest important moments in rugby games. The main indicators include referee's whistle that can suggest a

severe foul, try, conversion kick or other important events. This whistle is distinctive enough to be distinguished and thus useful for the study. The commentators' excited speech and crowd noise were also considered as they correlate with the excitement of the scene. However, the latter was found not very representative due to the continuous cheering from the crowd along the game. Their approach consists of five Gaussian Mixture Models, each to classify audio segments as a speech, excited speech, unexcited speech, whistle and other acoustic events. Firstly, the audio segments are classified as 'speech' or 'non-speech'. Secondly, the segments classified as 'speech' are classified in a second stage as 'excited speech' or 'unexcited speech' while the 'non-speech' segments as 'whistle' or 'others'. Finally, in order to collect the interesting moments, the occurrences of 'excited speech' and 'whistle' are stored in a buffer with their respective timestamp. The authors claimed that the approach reached accuracies of 97.06% and 93.41% for detecting try events and highlight scenes.

Other studies also leveraged visual and textual cues for sports highlight detection. In the paper "Automatic Curation of Sports Highlights Using Multimodal Excitement Features", Michele Merler et al. present a highlights generator system for tennis and golf taking advantage of players' reactions, expressions, crowd cheering and commentator's speech and game analytics [51]. Players' reactions such as high-fives and fist pumps are some of the indicators the researchers studied to extract interesting moments. To do so, classifiers based on the CNN models VGG-16 and ResNet-50 were used to detect these celebrations. These models are pretrained on ImageNet. However, since ImageNet does not contain a category of a person celebrating, it was necessary to train the models on new data from 2016 Masters, Wimbledon and US Open videos. Apart from the gestured reactions, facial expressions were also found to be valuable for the researchers. A single shot multibox detector was used

for the face emotion recognition task, classifying faces as aggressive, tense, smiling and neutral.

Crowd cheering and commentator excitement detection was approached using SoundNet audio features to model excitement, and a linear supporting vector machines classifier. Text-based commentator excitement was implemented using a dictionary of 60 exciting expressions and their excitement score. The Watson IBM speech-to-text service was used in order to extract the transcript from the commentators. To calculate the excitement score, the aggregate of the expressions' scores is computed.

The researchers also found it useful to use game analytics. Live information about tennis points provided by statisticians are used to identify key points such as match points and set points. These points are more valuable and are usually important for tennis highlights.

In the study “An intelligent strategy for the automatic detection of highlights in tennis video recordings,” the audio waves are analyzed for detecting patterns in the sound that are indicative of four types of events: serve, rally, break and replay [38]. A serve event is characterized by non-speech from commentators and a silent crowd. During a rally, the audio would consist of a sequence of speech with a low to medium level of noise from the crowd. A loud crowd with no speech would suggest a break. Lastly, both loud crowd and speech is an indicative for a replay.

The structure of a tennis broadcast is shown in Figure 3.1. Using the information from audio patterns and techniques to extract information from visual cues, the approach processes the broadcast to locate serve-rally-break events and includes them in the highlights.

As discussed in Chapter 1.2, professional tennis tournaments have recently started to use AI based highlights generators. Big tech companies such as IBM and Infosys are now providing some of the biggest tennis events such as Wimbledon

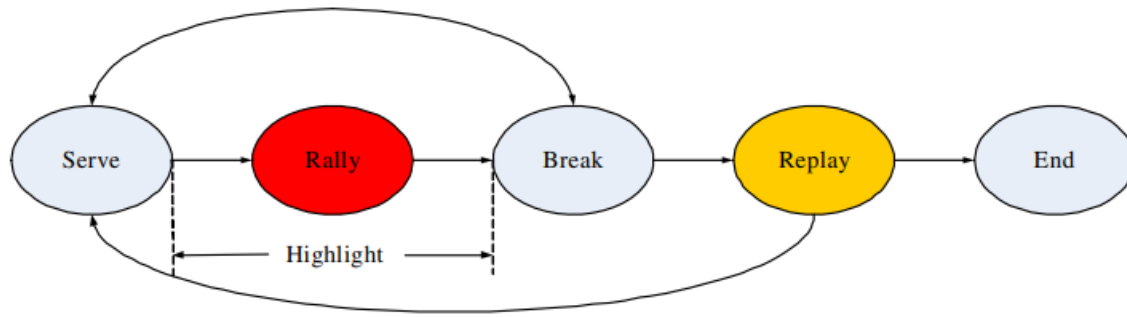


Figure 3.1: Structure of a tennis broadcast [38]

and Roland Garros with solutions [53, 47]. However, their software is continuously being improved and is not open to the public.

CHAPTER 4

COMPUTER VISION TECHNIQUES: METHODOLOGY

As introduced in Chapter 2.1, computer vision is a subfield of AI which trains computers to interpret and perceive meaningful information from the visual world. This chapter discusses the applications of computer vision in camera shot detection and player/court detection, as well as project implementations of Tracknet, YOLO and Hough transform.

4.1 SHOT DETECTION

One of the key tasks in this project is to accurately identify the timestamp of the point boundaries. Unlike other sports such as soccer where it is possible to extract a scene of a goal just by going back by a fixed amount of time, in tennis it would not make sense to cut a part of a point and include it in the highlights. Proper tennis highlights consist of the most interesting points from start to end.

Extracting the time intervals of the points is not an easy task. Previous studies took advantage of the input from court-side statisticians who actively annotate tennis matches in real time [51]. However, as the goal of this project is to to automatize this process without relying on manual labor, it is necessary to find an approach to solve this task. Conveniently, although official tennis matches are streamed with multiple cameras, the points are always filmed with the same satellite camera from start to end, whereas the other cameras are used to zoom in the crowd

and the players between points. It is possible to take advantage of this by detecting all the timestamps of the camera shot changes, and then classifying each as point in play or not in play.

The first part of the point boundary extraction is camera shot detection. This can be done with PySceneDetect [12], a command-line application and a Python library written on top of OpenCV for detecting shot changes in videos. PySceneDetect is an open source tool, and therefore provides a detailed documentation of the different algorithms that are available.

The PySceneDetect documentation discusses three important shot detection algorithms that are implemented: context-aware detector, adaptive content detector and threshold detector. Context-aware detector is an algorithm that detects jump cuts in the video. A jump cut is an editing technique that cuts between two sequential shots where the camera position has minimal change, but the subjects can move giving the appearance of jumping around frames. This generates the effect of moving forward through time. The content-aware detector can tell if two adjacent frames belong in the same scene. It takes the average difference across color channels from frame to frame, triggering a scene change when it exceeds a threshold.

Secondly, the adaptive content detector works in a similar way to the content-aware detectors, but uses a rolling average of adjacent frame changes in order to reduce false detections where there is a fast camera motion.

The threshold-based scene detector is the most traditional scene detection approach. The method consists of comparing the intensity/brightness of a frame with a threshold. Every time this value crosses the threshold, the detector considers it as a scene cut. The intensity value is calculated with RGB, computing the average value of each of the three channels for each pixel in the frame. The average pixel value is a floating-point number ranging from 0.0 to 255.0.

4.2 YOLO (YOU ONLY LOOK ONCE)

YOLO [62] is a fast and accurate object detection algorithm introduced by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in 2016 . It leverages convolutional neural networks to provide real time object detection, and it has been used in different applications such as autonomous driving, wildlife and security by detecting traffic signals, people, parking meters, and animals.

Prior to YOLO, detection systems repurposed classifiers or localizers to perform detection. The models were applied to an image at multiple locations and scales while giving high scores to regions of the image considered detected objects. Previous approaches such as deformable part models and regional convolutional neural networks were able to perform with great accuracy but did not excel in speed.

With YOLO, it was possible to overcome the processing speed limit by implementing an approach with a single CNN. The name of the algorithm YOLO is an abbreviation for the term ‘You Only Look Once’, suggesting that it requires only a single forward propagation the network in order to detect various objects in a picture in real time.



Figure 4.1: Demonstration of the input image being divided into an $S \times S$ grid [31]

As mentioned beforehand, YOLO does not search for interesting regions in

the image that may contain objects. Instead, the idea of the algorithm consists of splitting the image into a grid of dimension $S \times S$, as shown in Figure 4.1. A cell is responsible for detecting an object if the center of the object resides in it. Each grid cell forecasts B bounding boxes, provides their confidence scores, and predicts the C class probabilities. Each cell is typically responsible for predicting 5 bounding boxes due to the possibility of containing more than one objects. Each bounding box is composed of 5 predicted values which range from 0.0 to 1.0: x , y , w , h , and confidence. The variables x and y represent the center of the box relative to the cell, whereas w and h correspond to the width and height. The probability or confidence for the box to contain an object is defined as the equation:

$$\Pr(\text{Class}_i \mid \text{Object}) * \Pr(\text{Object}) * IOU_{pred}^{truth} = \Pr(\text{Class}_i) * IOU_{pred}^{truth} \quad (4.1)$$

$\Pr(\text{Object})$ is the confidence that an object was detected regardless the class. The term $\Pr(\text{Class}_i \mid \text{Object})$ is a conditional probability for each of the label classes given there is an object in the cell. IOU refers to intersection over union, and is a measure that compares predicted and ground truth bounding boxes. As shown in Figure 4.2, the intersection is the overlap between the boxes, whereas the union is their combined area. This value ranges from 0 to 1, where higher values represent high accuracy as the predicted and ground truth boxes are close.

Every grid cell from the $S \times S$ image grid predicts B bounding boxes and C class probabilities, outputting predictions of the shape $C + B * 5$. B is multiplied by 5 because it includes x , y , w , h , and confidence for each box. The overall prediction of the model in a $S \times S$ grid is a tensor of shape $S \times S \times (B * 5 + C)$.

Figure 4.3 depicts the YOLO system. It should be noted that while each cell predicts B bounding boxes and their confidence scores, it predicts only one class. Due to this limitation, the model fails to classify multiple objects of different classes

in one grid cell. However, a grid with very small cells can minimize this issue as objects are less likely to correspond to the same cell.

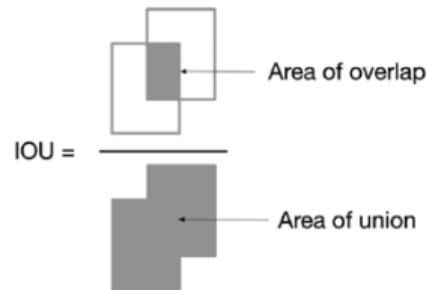


Figure 4.2: A visual explanation of the IOU metric [52]

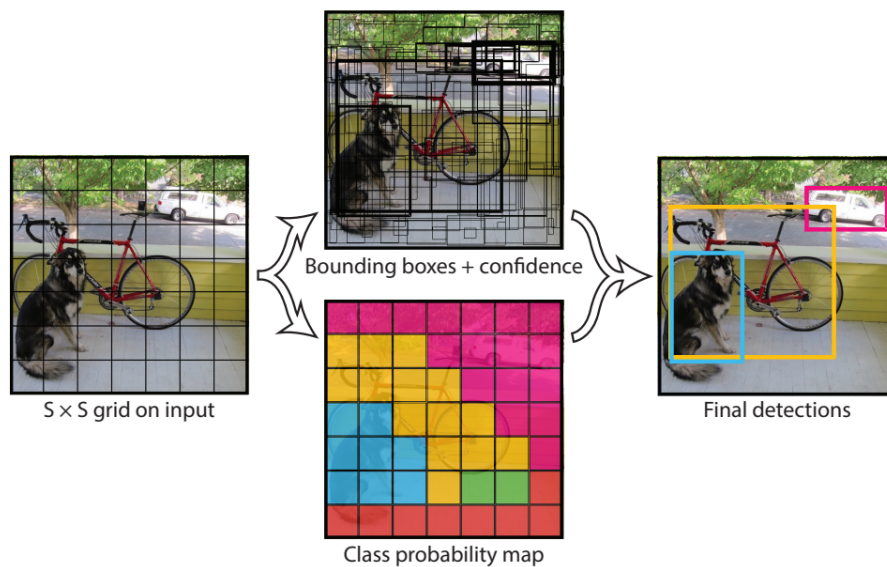


Figure 4.3: YOLO system to produce final detection results [62]. Each cell predicts B bounding boxes and C class probabilities. The highest class probability will determine the class in the class probability map.

Another issue found by researchers is detecting one object more than once in an image. Ideally, for each object in the image, there must be a single bounding box. A way to handle this is applying non-maximal suppression. YOLO leverages this technique to select the best bounding box from the multiple predicted bounding boxes. The idea of non-maximal suppression is to take the boxes with maximum

probability and remove the boxes close to it with lower probabilities, removing detections that correspond to the same object.

4.2.1 YOLOv3

After the introduction of the model in 2016, there have been new versions that increased the overall efficiency. The first version introduced the general architecture, and the second one substantially increased the accuracy. The current and most relevant version to this project is the third: YOLOv3. In this iteration, the design was improved with methods such as multiscale prediction and bounding box prediction, taking advantage of logistic regression [61].

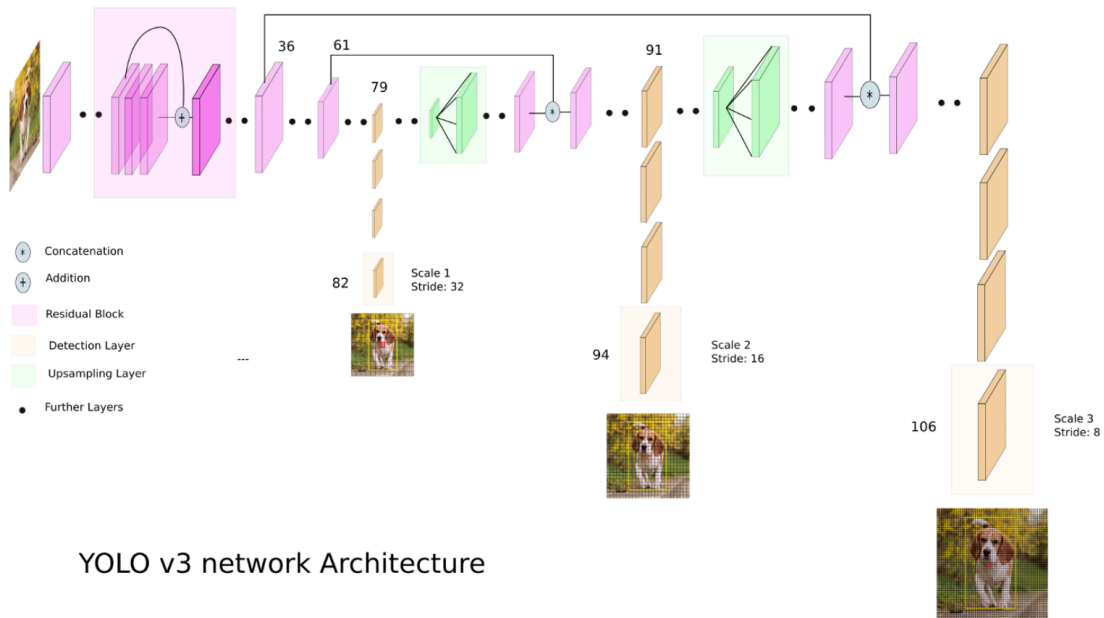


Figure 4.4: YOLOv3 Architecture [13]

The architecture of YOLOv3 is shown in Figure 4.4. In this model, the first 53 layers take care of the training task and then the other 53 layers of the detection task, accumulating to a total of a 106-layer CNN architecture. One of the features of this version is that the detections are made at three different scales in layers 82, 94 and 106. The convolutional layers contain kernels of size 1x1 that are applied three

times along the model on feature maps of three different sizes. Moreover, when performing detections at the three scales, downsampling is used for reducing the spatial resolution while keeping the same image representations.

Another feature of this third iteration of YOLO is the change of activation function. Previously, the softmax activation function was used to determine the classes of the objects. However, this approach assumes the classes are mutually exclusive. This assumption may fail if there is an object that belongs to two categories such as “car” and “vehicle”. To tackle this issue, the authors decided to rely on logistic regression to get a score for each class and be able to perform multilabel classification.

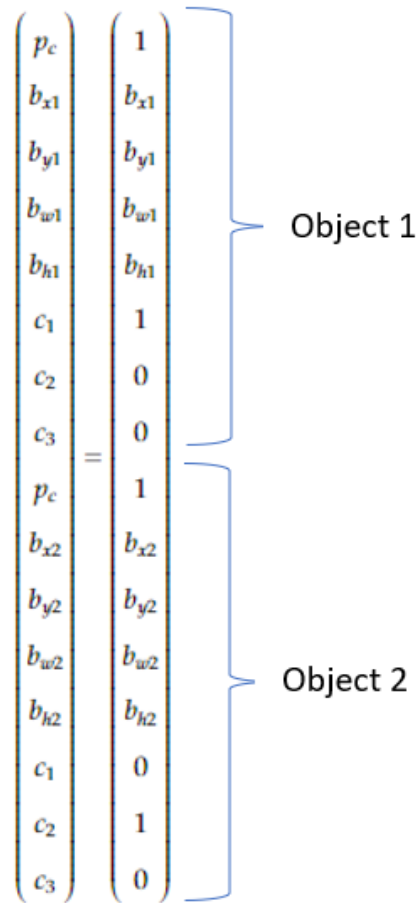


Figure 4.5: Illustration of anchor boxes being used to detect two objects in the same grid cell.

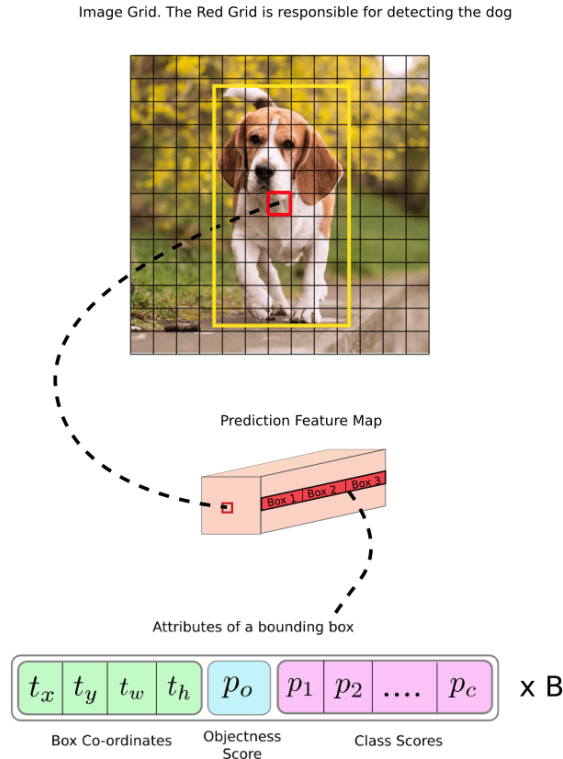


Figure 4.6: YOLOv3 steps. The red cell is responsible for detecting the dog [13].

This iterations of YOLO also tackles the limitation for detecting objects that are too close to each other. This was solved with anchor boxes, which allow to detect multiple objects centered in one grid cell. The approach consists of adding dimensions to the predictions, so that the class prediction is moved from the cell level to the bounding box level.

Previously, the output vector prediction that represents the grid could only predict one class. With anchor boxes, the predictions for objects of different classes are concatenated. Figure 4.5 shows how two anchor boxes can be used to detect two objects in the same grid cell. In this example, it can be seen that the dimension of the vector is doubled to contain boundary boxes with different class probabilities. Here, the first half of the vector contains the object probability p_c , the box coordinates b_{x1} , b_{y1} , b_{w1} and b_{h1} and the class probabilities c_1, c_2, c_3 for object 1, and the other half contains b_{x2} , b_{y2} , b_{w2} , b_{h2} , and the object and class probabilities for object 2.

Figure 4.6 shows the steps of YOLOv3. Here, the red cell is responsible for detecting the dog. It can also be seen that YOLOv3 uses three anchor boxes are used when performing detections. Each detection contains the coordinates of the predicted bounding box, the object probability and the probability for each class. Therefore, they have a shape of $B * (5 + C)$.

4.3 TRACKNET

When analyzing game strategies and player's performance in different sports, one of the factors that give us the most information is the trajectory of the ball. There has been previous work to track objects in sports applications. However, being able to recognize small objects that move with a high velocity and which can be blurry across frames is a more difficult task. In this section, TrackNet, a convolutional neural network for tracking high speed and tiny objects, is introduced [39].

Tracknet was presented in a 2019 paper titled "TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications," by Yu-Chuan Huang et al [39]. This model is a heatmap-based deep learning network that is able to take an input frame and recognize a tennis ball. Moreover, this model can also learn flying patterns from consecutive frames, which can be used to estimate the location of the ball even in those frames where it is covered by any object. Figure 4.7 showcases how TrackNet detects the ball in a single frame.

In order to train the model, the authors manually labeled more than 20000 frames by hand with the following four values:

1. Frame Name.
2. Visibility Class: The number 0 indicates the ball is outside the frame, 1 is within the frame and 2 for when it is in the frame but it is occluded.

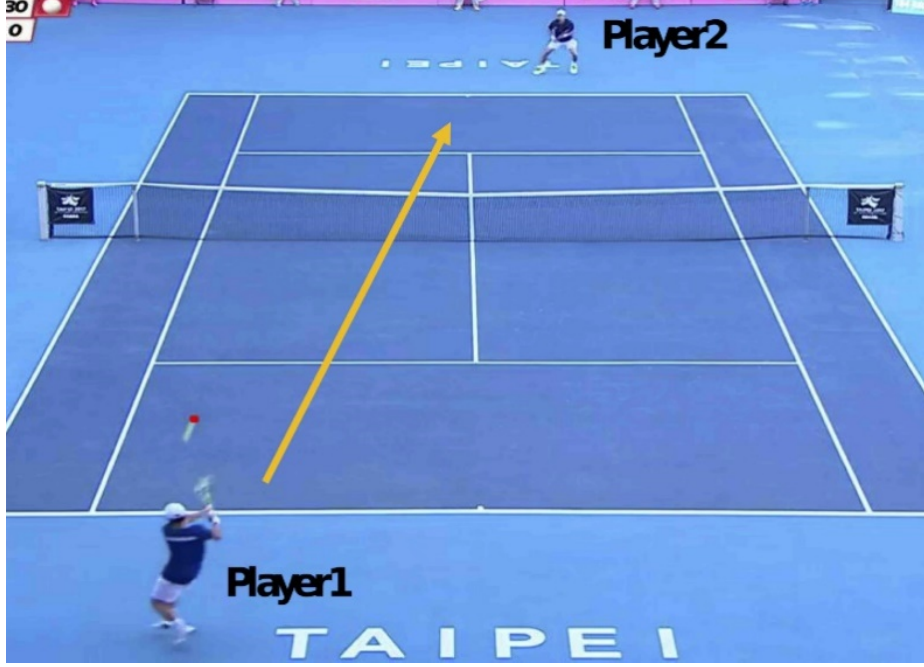


Figure 4.7: Output frame showing the tracked ball [39].

3. (X,Y): The coordinates of the tennis ball. In the cases where it is blurry due to high velocity, the model considers the last position of its trace.
4. Trajectory Pattern: It classifies the movement in flying (0), hitting (1) and bouncing (2).

Tracknet was trained to output heatmaps with the same resolution as the input frames. The ground truth of the heatmap is an amplified 2D Gaussian distribution located at the center of the tennis ball, as visualized in Figure 4.8. The variance of the Gaussian distribution refers to the diameter of tennis ball images, which is assumed to be 10 pixels. The Gaussian heatmap function is defined as [39]:

$$G(x, y) = \lfloor \left(\frac{1}{2\pi\sigma^2} e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}} \right) (2\pi\sigma^2 \cdot 255) \rfloor \quad (4.2)$$

Where the left term represents the Gaussian distribution centered at the ball center (x_0, y_0) with variance σ^2 and the right scales the heatmap to the range of

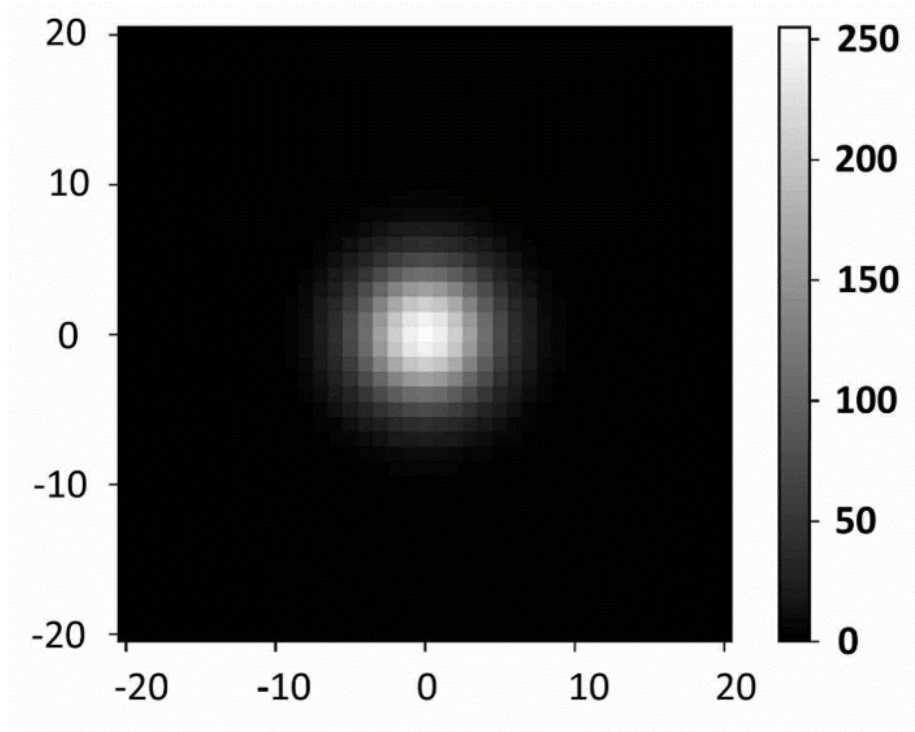


Figure 4.8: Visualization of a zoomed in ball heatmap [39].

[0,255]. In their implementation, the variance σ^2 was set to 10 since the average ball radius is about 5 pixels.

Tracknet is composed of two neural networks. The first is a CNN inspired in the model "VGG16" [67], which is highlighted in Figure 4.9. The first 13 layers of Tracknet refer to the design of the first 13 layers of VGG16 for object classification. In VGG16, an image of a default size of 224 x 224 passes through a stack of convolutional layers where filters of 3x3 and 1x1 and a stride of 1 pixel are applied. Pooling is done with five max-pooling layers over a 2x2 filter with stride 2. Then, two convolutional layers of 4096 and 1000 channels perform the classification task. The final layer consists of a soft-max layer. In total, these compose 14 layers and are used for the feature extraction portion.

The second model Tracknet uses for the 14-24 layers is the deconvolutional neural network "DeconvNet" [55], and is used for the spatial localization. A deconvolutional neural network is a CNN that works in a reversed process. The architecture for

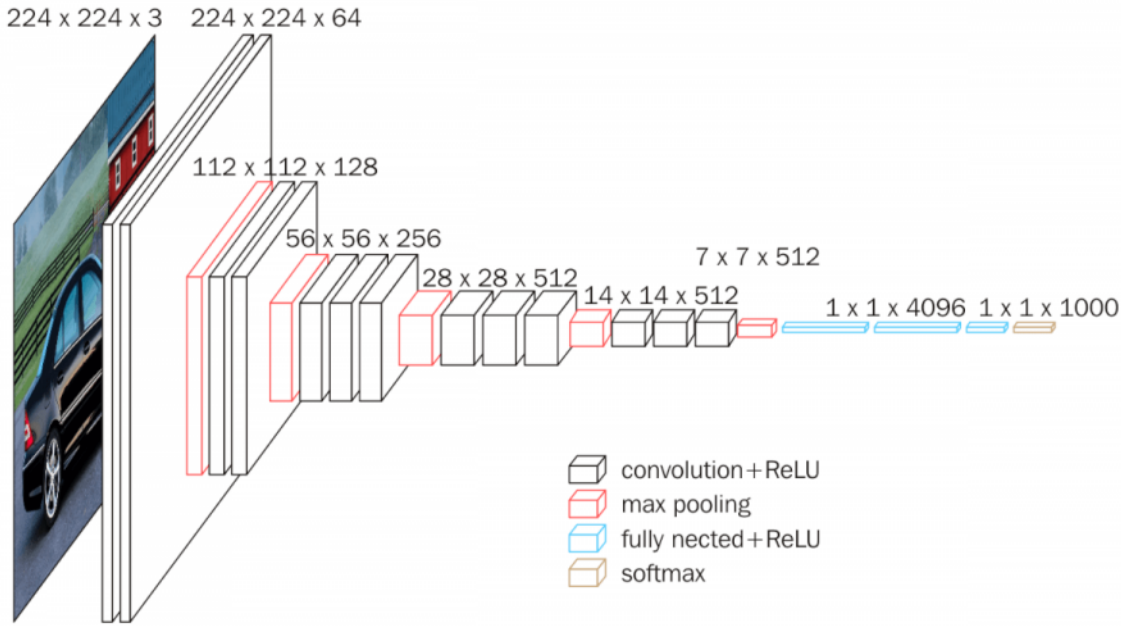


Figure 4.9: VGG16 Architecture [72]

this model is showcased in Figure 4.11. DeconvNet consists of deconvolution and unpooling layers. Unpooling layers perform the reverse operation of pooling and reconstruct the original structure of the input object. This is useful to recover the spatial information that is lost previously in the pooling layer of VGG16, which can be important for precise localization. The output of this layer is an enlarged activation map. To implement the unpooling operation, switch variables are used to record the locations of the maxima when performing max pooling, and place these maximum activations back to the original locations.

The deconvolutional layers are then used in order to densify this output though convolution-like operations with filters. They differ from convolutional layers because they associate a single input activation with multiple outputs, instead of multiple input activations to a single activation. Illustrations for the two operations are shown in Figure 4.10.

Figure 4.12 shows the architecture of Tracknet. After these two networks, a softmax layer outputs the predicted probability-like heatmap of a tennis ball with

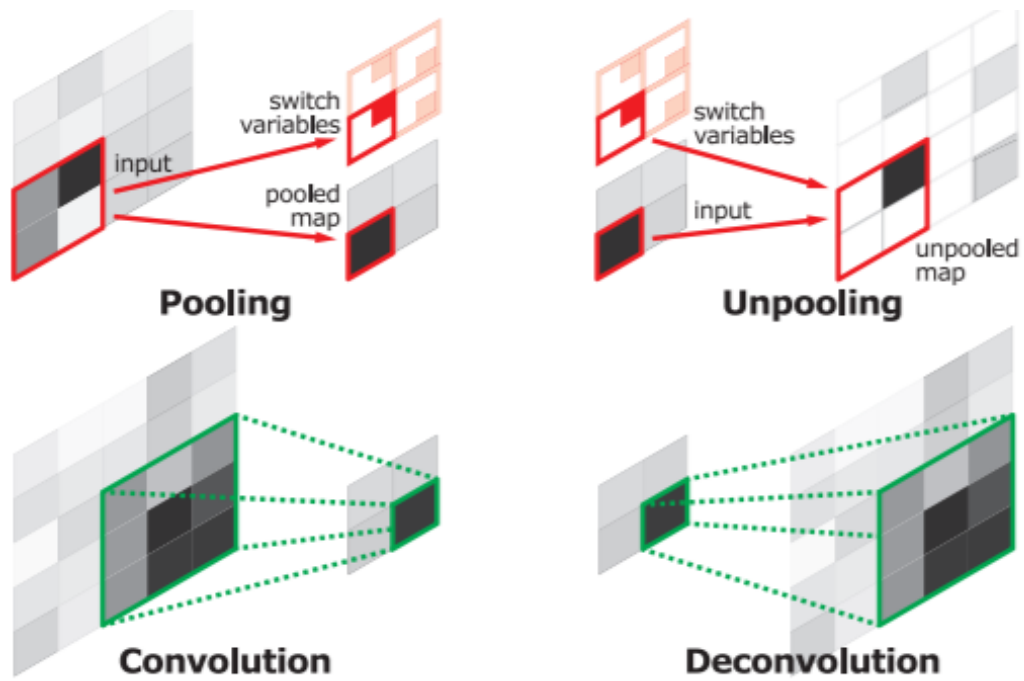


Figure 4.10: Image showing how unpooling and deconvolution operations are the inverse of pooling and convolution [55].

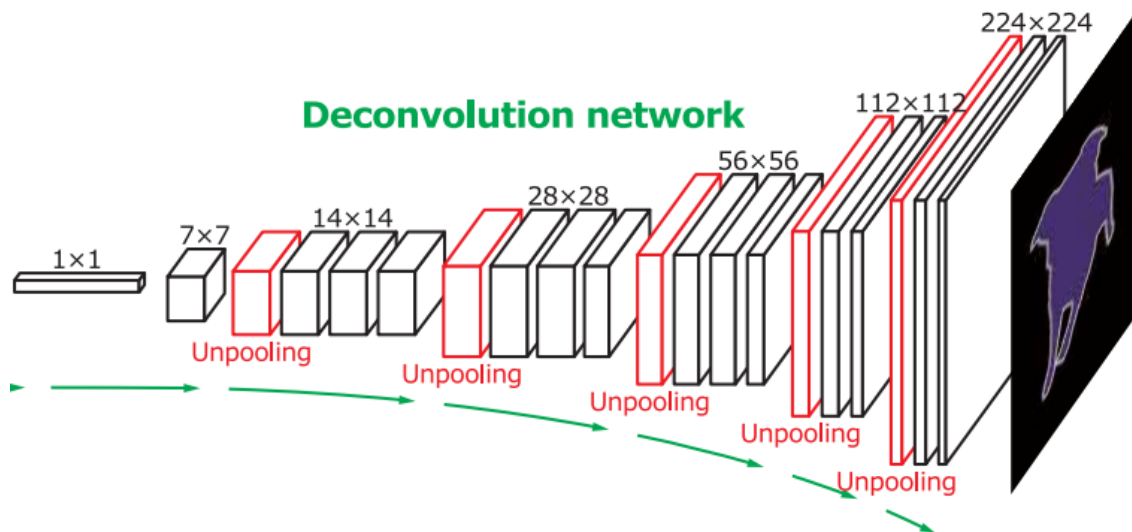


Figure 4.11: DeconvNet Architecture [55]

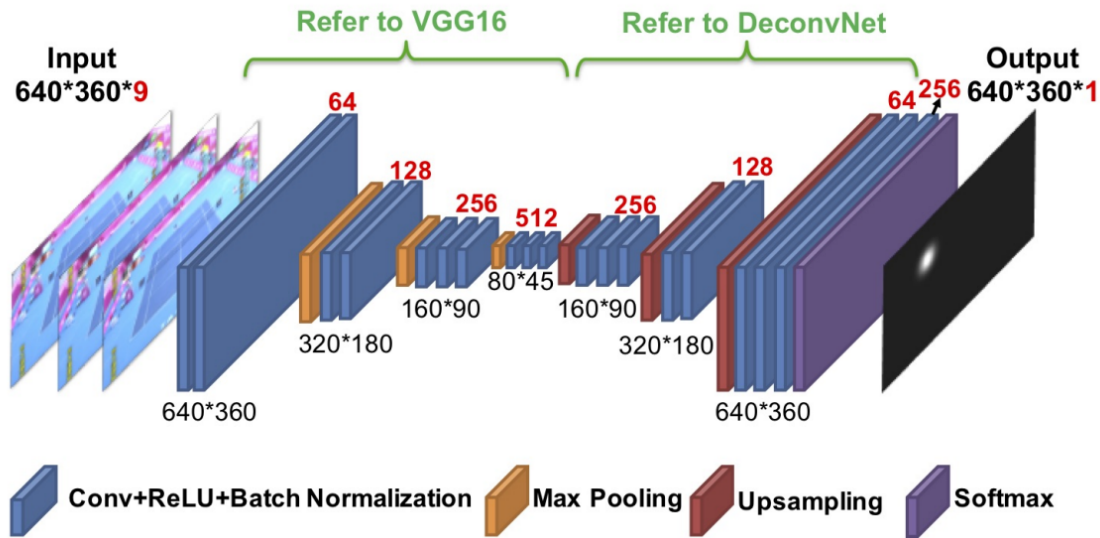


Figure 4.12: Tracknet Architecture [39]

the same dimension as the input where each pixel is a grayscale value in the range $[0, 255]$. Finally, the heatmap is converted to a binary heatmap using a threshold of 128.

The researchers tested three variations of the model to evaluate the performance of TrackNet. Tracknet Model I takes of a single frame input, TrackNet II takes three consecutive input frames. The comparison showed great performance improvement over conventional algorithms, with model II performing even better than model I. Moreover, the authors found that Tracknet outperforms other conventional image processing algorithms.

4.4 HOUGH TRANSFORM

Hough transform is technique from computer vision developed by Paul V.C. Hough to detect complex shapes in an image such as lines, circles and ellipses. The main idea in this technique is that each coordinate point contributes to a globally consistent solution [54].

In image space, a line can be represented using the slope intercept form

$$y = ax + b \quad (4.3)$$

and also the polar coordinate form

$$x \cos \theta + y \sin \theta = \rho \quad (4.4)$$

where a is the slope, b is the y-intercept of the line, ρ is the radial distance from the origin and θ is the counterclockwise angle from the x-axis.

The Hough space is a 2D plane that plots either a vs b or θ vs ρ . In this representation, each line of the image space is represented by a dot. The intuition of Hough transform is to scan the image space and get a, b or ρ, θ for each of the points. Every point that conforms a line is mapped to the same point in the Hough space, incrementing its accumulator. A high count for a point in the Hough space suggests the existence of a line [54, 45].

Figure 4.13 shows simple examples of mappings from the image to the Hough space. A line on the image space produces a point (a, b) or (θ, ρ) on the Hough space. However, a point (x_i, y_i) in the image space have a family of lines that go through it, so it produces a line in the Hough space of the form $b = ax_i + y_i$, or a cosine curve $\rho = x_1 \cos(\theta) + y_1 \sin(\theta)$. The idea of Hough transform is that if points in the image space that lay on the same line, their corresponding lines or curves in the Hough

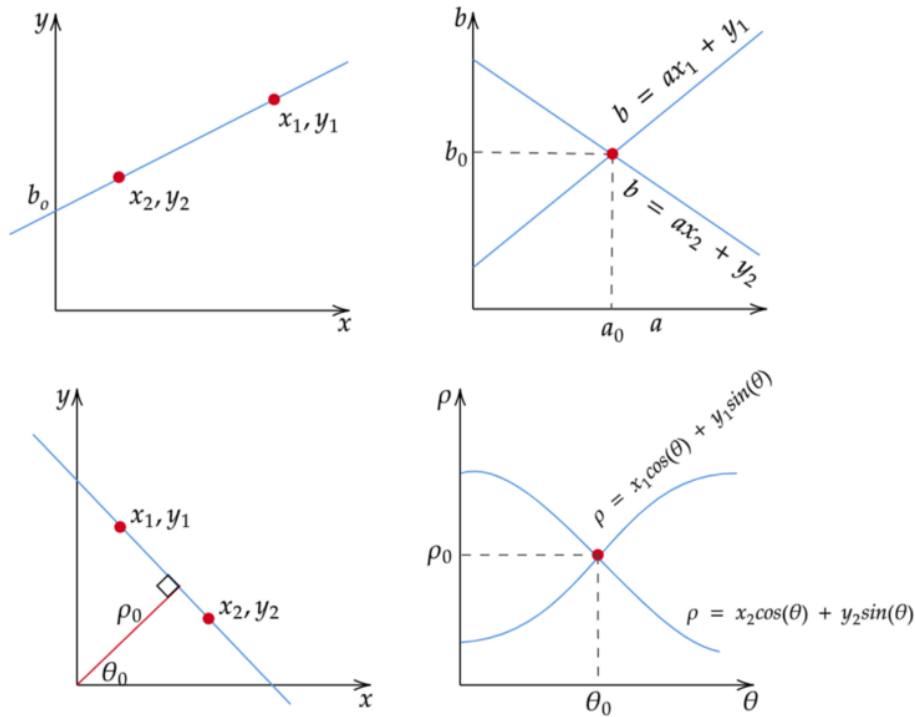


Figure 4.13: Image Space and Hough Space. The top images correspond to rectangular coordinates and the bottom images to polar coordinates [45].

space intersect in the same point. The algorithm keeps track of the intersection between curves of every point in the image. If the number of intersections is above some threshold, then it declares it as a line. The graphs in the top of 4.13 show a line mapped to a point in the Hough space in rectangular coordinates, where all the points that lay on it produce lines that intersect in the same point. The graphs below show a line mapped in polar coordinates, and the points in it produce cosine curves that also intersect in a point from the Hough space.

4.4.1 THE HOUGH ALGORITHM

The following steps illustrate the Hough algorithm. Figure 4.14 shows an example of Hough transform using *HoughLinesP* from OpenCV.

1. Lets consider the image space using polar coordinates. The first step is to

determine the range of ρ and θ . This step is important because there should be a finite number of possible values.

2. Create a 2-dimensional array to keep track of the accumulators in the Hough Space.
3. Perform edge detection in the input image with an algorithm of choice. Edge detection algorithms detect where brightness changes drastically.
4. For each pixel on the edge image, check whether it is an edge pixel. If this is the case, loop through all possible values of θ , calculate ρ and increment the corresponding accumulator.
5. Loop through the accumulator array and select the values greater than a threshold.



Figure 4.14: Illustration the results of Hough transform on an image. The left picture consists of the input image, the middle is the result of edge detection, and the right shows the lines detected. [42]

4.5 CV TECHNIQUES IMPLEMENTATION

After applying shot detection to the input video of a full tennis match, a list of the timestamps of each "scene" is obtained. The next step consists of classifying each of those scenes as either point in play or not in play. In our study we use TrackNet for detecting the ball, YOLOv3 for the players and Hough transform for the court. An

illustration of the target camera shots that correspond to tennis points is showcased in Figure 4.15.

The camera scenes where the three of these are detected represent the points with great accuracy. However, there can be some false positives such as replays that do not correspond to points and should be discarded. These redundant points are going to be handled in the Audio Analysis section.

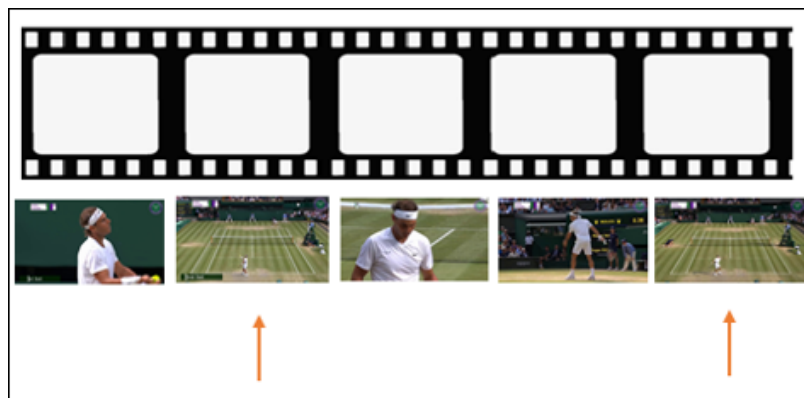


Figure 4.15: Illustration from camera scenes in tennis match and the target points.

The TrackNet project was cloned from the official repository. However, it was necessary to make some adjustments in order to fit the needs of our project and optimize time and efficiency. TrackNet was programmed to analyze videos frame by frame and then create an output video displaying the tracking of the ball. The second part is not only time consuming, but also unnecessary for the highlights generator as our only interest is to know whether or not a ball is detected. Therefore, this portion was removed. Moreover, analyzing frame by frame would be necessary to visualize the tracking. This would run the model too many times, which is expensive and requires hours of processing. Considering the ball is blurry and can be covered in some frames, the decision was to choose 5 frames evenly spread in each camera scene, run the model for each, and select the scenes where there was at least one detection.

In order to leverage YOLOv3 for player detection, the open-source github repository "Track tennis players and the ball" was chosen [6]. The reason is that they took the time to apply the models and make all the necessary adjustments so that it can be ran in an official tennis match. The process of adaptation is very tedious because there is a lot of processing needed to handle details such as discriminating the players from other persons present in the frame that are not players such as the judges and ball kids. To implement YOLOv3, the function `cv2.dnn.readNet(yoloweights, yoloconfig)` from OpenCV is used. To read the net, this function takes two files of the model architecture and weights as the input.

To perform court detection with Hough Transform, the github repository "tennis-tracking" [4] by ArtLabs was used. In this project, the Hough Line Transform `cv2.HoughLinesP()` from OpenCV is used to detect the lines in the court. The lines detected are classified as horizontal and vertical, and then the lines that "belong" to the same line are merged.

Just like Tracknet, the main objective of these project is to display the tracking of the objects, so the model is run frame by frame and an output video is also produced to show the bounding boxes in real time. Therefore, the same process was applied to adapt it to fit the highlights generator, with the difference that only one frame was chosen to be processed per scene due to the bigger and clearer target objects. Figure 4.16 shows an example of what the models achieve when ran on each frame.

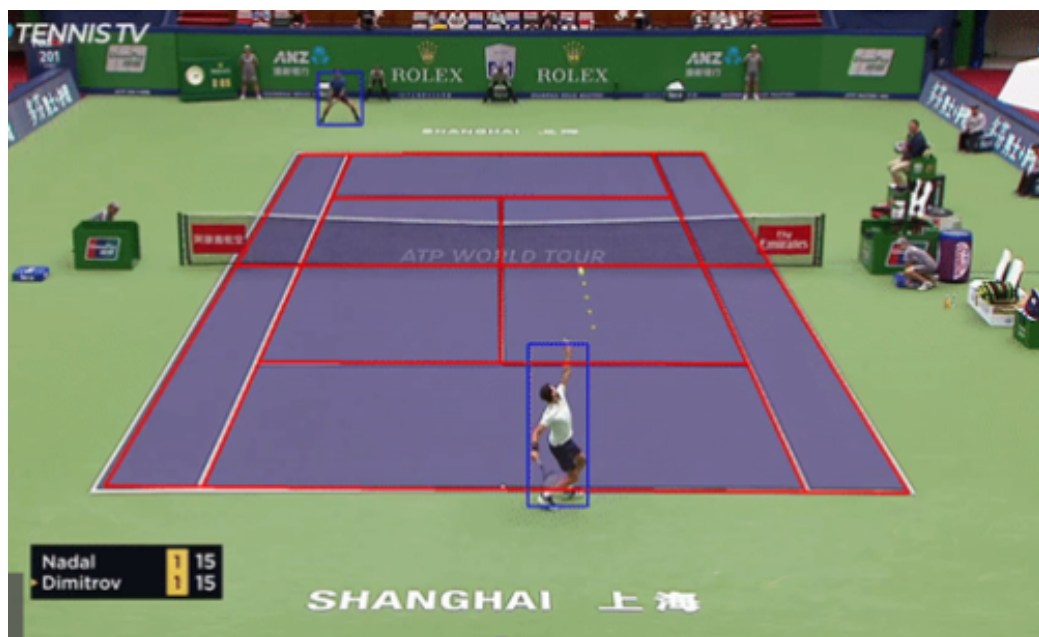


Figure 4.16: Visualization of court, ball and players detection in a single frame from the "Tennis-tracking" project [4].

CHAPTER 5

AUDIO ANALYSIS TECHNIQUES: METHODOLOGY

Cheers from fans and tone increases from commentators are very useful to find the exciting moments in sports events, and tennis is not an exception. One approach that has been applied in previous research is to build a model using machine learning [51]. This process would require gathering a lot of audio streams of tennis matches and manually annotating them to construct classifiers for crowd-cheering and commentator tone excitement. However, in this project we will take a different approach which is simpler and more straightforward, while also being capable of fulfilling our goal. This approach consists of analyzing the audio waves to identify peaks in the audio, which indicates when the crowd cheers or commentators are excited the most. It has been inspired by a project by Prateek Karkare for soccer games [43].

The first step is to extract the audio from the video in .wav format. To do so, the library FFmpeg is used [19]. FFmpeg is a free and open source software project for handling video and audio streams. It is used to analyze, encode, decode, process and modify audiovisual data. It has been managed and maintained by Michael Niedermayer since 2004, and a large number of developers have been contributing to the project on a daily basis. FFmpeg is part of the workflow of many other software projects. Its libraries are a core part of software media players such as VLC, and it has been included in core processing for YouTube and Bilibili.

Nowadays, most of the audio files that can be downloaded from internet are

stereo due to the better quality. Stereo audio consists of two audio channels. However, in order to extract the exciting parts, our study requires the audio to be converted into a mono channel audio by averaging the two audio channels of the .wav file.

A moving average filter is ran to reduce noise by removing high frequency elements. A moving average filter is a basic technique that consists of taking the average of the last “M” amount of entries in the signal and averaging them to produce the output. The only parameter that can be controlled is the window size, which is the amount of previous signal entries that can be averaged together. The key is to choose the right window size, and can be done by trial and error. If it is too small, the signal will still end up being very noisy. Conversely, if it is too large, the signal will lose important information [69].

The attribute that suggests the exciting segments of the audio is the intensity. The intensity of a wave [56] is the power per unit area carried by the wave. Power is the rate at which energy is transfer by the wave. The intensity I has the form

$$I = \frac{P}{A} \quad (5.1)$$

where P is the power through an area A . The intensity of the sound depends on its pressure amplitude. The relationship is

$$I = \frac{(\Delta p)^2}{2\rho v_w} \quad (5.2)$$

where ρ is the density of the material, v is the speed of the sound through the medium and Δp is half the difference between the maximum and minimum pressure in the sound wave.

From equation 5.2, we have that the intensity of the sound is proportional to the

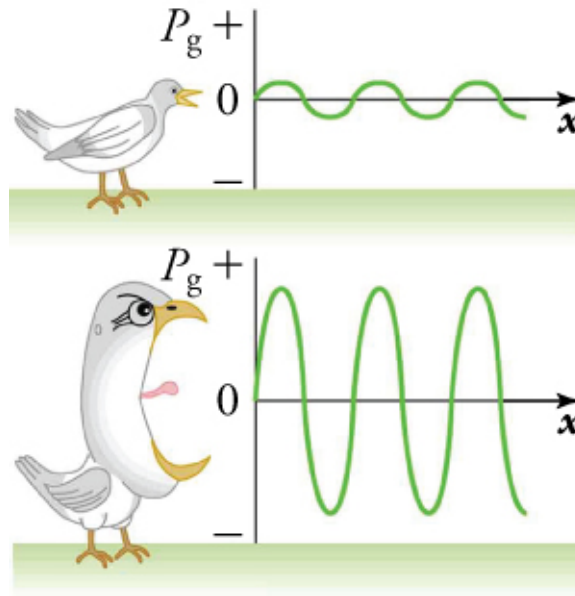


Figure 5.1: Graphs of the pressures in two sound waves of different intensities. The more intense sound is produced by a source, the larger the amplitude of the oscillations [56].

square of its amplitude. Therefore, the plot of the square of the amplitude over time tells the level of excitement along the match, as illustrated in Figures 5.1 and 5.2.

In tennis, points always need to be played in silence, and thus cheering happens right after each point finishes. The peaks in the Figure 5.2 represent the biggest reactions to points, and it can be inferred that the higher the peak, the more exciting the point. The strategy for choosing the best points is to take advantage of the point

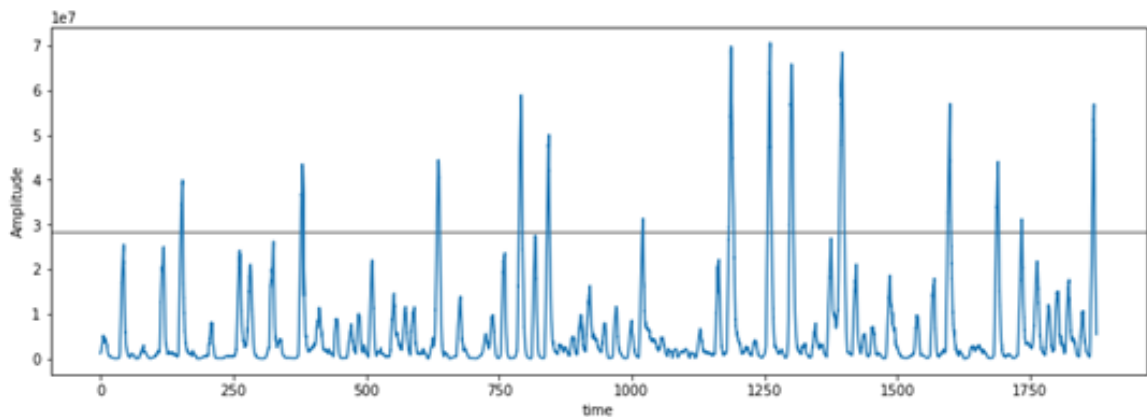


Figure 5.2: Square of the amplitude over time

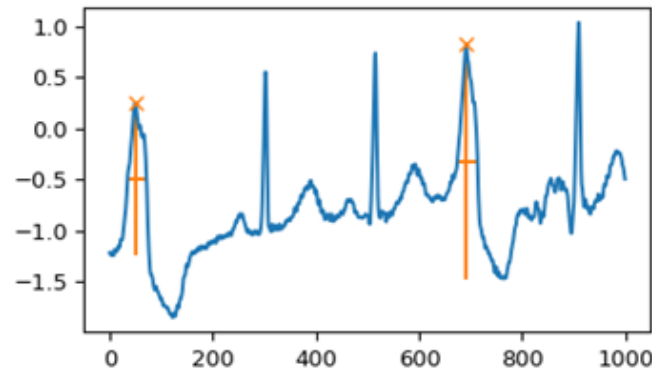


Figure 5.3: Peak Detection

boundaries obtained with the computer vision techniques and select the points that happen just before the highest peaks in the graph.

The peak detection algorithm from SciPy was chosen in order to find the peaks in the graph. The algorithm `scipy.signal.find_peaks` searches for local maxima based on simple value comparison of neighboring samples and returns peaks whose properties match optionally specified conditions for their height, prominence, width, threshold and horizontal distance. Figure 5.3 shows how this algorithm performs in a graph.

5.1 APPLICATION FOR IMPROVED POINT DETECTION

This audio analysis approach can also be applied to improve point detection and discard camera shots of redundant points. As explained in Chapter 4, the accuracy of points detection with the computer vision techniques was almost perfect. Some of the cases they cannot handle are replays, as they are points in play that are duplicates. Other redundant scenes such as missed serves are also classified as

points, but do not correspond to actual points. Since highlights are meant to be short and interesting, we do not include these scenes.

The audio peaks are useful to solve this problem. Basically, we have two identical scenes that show the same point. However, what differentiate them is the audio. The replays are shown always after the points, and the audio only consists of the commentators' description of them. By the time a replay is shown, the cheering from the crowd for that point is already over. Therefore, it is possible to compare the audio of each, classify as point the one that contains a peak in the volume at the end, while discarding the other.

CHAPTER 6

NATURAL LANGUAGE PROCESSING TECHNIQUES: METHODOLOGY

Audio analysis is used for tone-based excitement detection. However, this technique might not be sufficient just on its own some scenarios. For instance, there can be matches without a crowd for different reasons such as covid-19 restrictions. Moreover, in cases where one of the players plays in his own country, the crowd may be biased towards one side and not cheer enough for points the opponent wins. Therefore, natural language processing and sentiment analysis are used for text-based commentator excitement, searching for the most interesting points according to what the commentators say right after they finish.

6.1 IMPLEMENTATION

The approach for text-based commentator excitement consists of two parts. Firstly, it is necessary detect the comments from the commentators. Afterwards, we can apply a sentiment analysis model to search for positive sentiment.

6.1.1 SPEECH RECOGNITION WITH GOOGLE API

The Google Speech-to-Text API was chosen in order to extract the commentators' transcripts from the video. The API use different machine learning models to target

different types of audio sources. The most suitable for this project is the "Video" model, which can handle different speakers and lots of background noise [73].

Considering this is one of the premium models of a paid API, it is neither viable nor necessary to send the entire audio of the match. It can be expected that the key comments about each point happen right when they finish. Thus, our approach consists of segmenting the audio file with FFmpeg in chunks that start 3 seconds after the end of each point and last 13 seconds. This will encapsulate most of the important descriptions the commentators give while skipping those 3 seconds in which the crowd is too loud for the recognizer to detect the speech properly. Afterwards, a loop is performed to send each .wav file to the API and store the responses in a dictionary.

6.1.2 SENTIMENT ANALYZER

The first idea was to leverage a third party sentiment analyzer. Many models, including NLTK Vader [59], were tested for the purpose of our project. However, as the objective is to detect positive sentiment that is related to tennis, none of these could fulfill our needs. These models are pre-trained for general sentiment analysis without any specific domain, so they were not able to classify a phrase like "nice forehand" as positive. Therefore, it was necessary to implement a model with a specific domain for tennis matches.

As discussed in Chapter 2.4.1, approaches for a sentiment analyzer include machine learning, rule-based and hybrid systems. The first option requires lots of labeled data in the target domain in order to be trained. Therefore, a rule-based system is more suitable for this project. For our model, two lists of domain-specific words were created: one of nouns that are relevant to tennis matches such as "forehand" or "point", and one of adjectives of positive words that are used to give

emphasis to the nouns like "huge" and "brilliant". Table 6.1 shows a segment of the dictionaries that were manually created.

Table 6.1: Segment of the lists for the Sentiment Analyzer

Adjectives	Nouns
good	forehand
great	backhand
huge	serve
excellent	volley
brilliant	point
exceptional	return
superb	rally
best	effort

The objective of this system is to classify the comments of each point as positive or negative. The algorithm of the sentiment analyzer consists of analyzing each phrase and classify as positive the ones that pass one of the following two hand-craft rules: containing at least one noun and positive adjective, or only a positive adjective. Table 6.2 shows a few tests with input phrases and their respective results.

Table 6.2: Sentiment Analyzer input tests

Input	Result
this was a very good shot	pos
how did he come with that passing, outstanding	pos
his forehand could have been better	neu/neg
Federer won three matches in a row	neu/neg
wow	pos

Listing 6.1 shows a code snippet of the main logic of the algorithm. Here, we are analyzing the respective phrase for every scene by looping over the dictionary with all the API responses. Then, the punctuation is removed and each word is checked if it an adjective or a noun. Finally, the rules are applied and the positive detections are appended to the output dictionary.

```
1  # loop over the phrase of each point
2  for scene, sentence in sentences.items():
3      #remove punctuation
4      s = removePunctuation(sentence)
5      curNoun = ""
6      curAdj = ""
7      isAdj = False
8      isNoun = False
9
10     #loop over each word in the phrase
11     for word in s.split():
12         if word in adjs:
13             isAdj = True
14             curAdj = word
15
16         #handle plural words
17         if word in nouns or (word[:-1]) in nouns:
18             isNoun = True
19             curNoun = word
20
21     #case 1: adj + noun
22     if (isAdj and isNoun):
23         results[scene] = {"a+n": [curAdj, curNoun]}
24     #case 2: only adj
25     elif (isAdj):
26         results[scene] = {"a": [curAdj]}
```

Listing 6.1: Sentiment analyzer code snippet

CHAPTER 7

RESULTS AND FINAL SOFTWARE PIPELINE

All the techniques investigated above are not very useful on their own. It is when merged together that we can achieve an efficient approach for solving the highlight generation task for tennis matches.

7.1 POINTS DETECTION APPROACH AND RESULTS

As described in Chapter 4.5, after the first step of applying shot detection, the different models for ball (tracknet), court (Hough transform) and players detection (YOLOv3) are ran. The models were tested in the first set of three different tennis matches. These matches are all available for free in YouTube, and were uploaded by the official US Open tournament account. The matches consist of the 2021 US Open men's final ¹, women's final ² and women's semifinal ³. In this test, the extracted camera scenes by each technique were analyzed and compared to the ground truth lists of points that were manually classified. The purpose of this test is compare the performance of the techniques in terms of efficiency and time consumption.

An important note is to understand the trade-off between type 1 and type 2 errors. In statistics, type 1 errors correspond to false positives, which in our case means to classify a camera shot as a point, when it actually not a point. Conversely,

¹https://www.youtube.com/watch?v=yACtNdJaBso&ab_channel=USOpenTennisChampionships

²https://www.youtube.com/watch?v=F99Kz2eptqM&ab_channel=USOpenTennisChampionships

³https://www.youtube.com/watch?v=l-q5imOkqg&ab_channel=USOpenTennisChampionships

type 2 errors are false negatives. Here, a camera shot is not classified as a point, but it is actually a point. For our purpose, a type 2 error is much worse than a type 1 error. This is because if an actual point is missed in this step, it will be discarded. On the other hand, a false positive can be handled and discarded from the highlights afterwards. However, minimizing type 1 errors is also important.

The results for the men's final, women's final and semifinal are shown in Tables 7.1, 7.2 and 7.3. PySceneDetect was used to divide the videos into 281, 434 and 390 camera shots. Afterwards, the different techniques were independently applied to filter these camera shots. The three tables show the total camera shots after each method, as well as the execution time and the false negatives and positives.

From these tests, we found that court detection with Hough transform was very effective to minimize both false negatives and positives. Here, the few false positives happened due to specific scenarios such as replays and missed serves where the court is shown but there is no point in play. The number of false negatives for the three tests were at most 2, whereas the false positives were 19, 34 and 21.

Tracknet for ball detection also correctly identified the points in play with small numbers of false negatives. However, it did not perform as great as the court detector at filtering undesired camera shots and minimizing false positives. Tracknet resulted in unwanted detections due to being designed to track very tiny and fast objects.

Player detection showed very similar results to ball detection. The biggest limitation found was that although YOLOv3 detect players during the points, the algorithm get confused and perform unwanted detections. The implementation can differ players from ball kids and judges during points, however, the algorithm fails in scenes where the crowd is shown. To overcome this situation and differ tennis players from people, it would be necessary to train a new class for a tennis player.

The results from these tests show that the court detector is the most efficient in

terms of performance. The three executions of the court detector lasted between 13 and 14 minutes each using the cloud GPU from Google Colaboratory, which is very good considering that even faster results can be achieved with a high-end GPU. Therefore, court detection was the chosen model to be included in the final software pipeline.

Table 7.1: 2021 US Open men's final first set test

Technique	Camera Shots	False Neg	False Pos	Time ex
PySceneDetect	281	-	-	4m
Hough Transform	70	2	19	13m
Tracknet	103	8	58	9m
YOLOv3	178	1	126	8m
Audio Peaks	49	10	6	1m

Table 7.2: 2021 US Open women's final first set test

Technique	Camera Shots	False Neg	False Pos	Time ex
PySceneDetect	434	-	-	6m
Hough Transform	111	2	34	14m
Tracknet	206	4	131	13m
YOLOv3	177	9	107	12m
Audio Peaks	87	5	13	1m

Table 7.3: 2021 US Open women's semifinal first set test

Technique	Camera Shots	False Neg	False Pos	Time Ex
PySceneDetect	390	-	-	5m
Hough Transform	101	0	21	13m
Tracknet	232	0	152	12m
YOLOv3	145	13	78	10m
Audio Peaks	68	20	8	1m

7.1.1 HANDLING FALSE POSITIVES WITH AUDIO ANALYSIS

Having a list with all the point intervals is essential in our project to reduce the amount of requests to the google API and optimize resources. Therefore, even though false negatives were minimized, there are still false positives that need to be discarded from the list of points.

One way that showed high efficiency is to leverage the audio analysis technique of analyzing the audio peaks. The approach consists of removing all the scenes where the audio is very low from the last three seconds until ten seconds after they end. This way, false positives such as point replays and repeated let serves can be discarded. Also, other redundant points including double faults or missed returns will also be deleted, resulting in a concise list of potential great points to analyze and include in the output highlights.

7.1.2 TIME VS EFFICIENCY: AUDIO ANALYSIS VS COMPUTER VISION

One approach that was considered is to set aside the computer vision detection techniques and rely only on audio analysis. The reason is that while analyzing image data with deep learning models and hough transform can take up to half an hour for long matches, analyzing the audio waves is a matter of seconds. However, there are several reasons why the camera shot and court detection are essential for our approach.

Audio analysis can be used to get the highest audio peaks in the match. However, the issue that raises is how much to go back before those peaks. In sports like soccer, this would very easy because we can go back a fixed amount of time before a goal and it almost certain that we are going to grab the desired play. In our project, we need to know exactly the what the time interval of the point is to obtain these target moments. Therefore, shot detection cannot be set aside.

Another alternative that was thought of to optimize time was to use audio analysis and shot detection, discarding computer vision techniques. As Tables 7.1, 7.2 and 7.3 show, one of the approaches tested was to apply the audio analysis right after shot detection. Despite the short time executions, this technique was found not to be consistent enough to replace court detection. Here, even though we have all the scenes that come before an audio peak, it is not possible to efficiently select the actual point. For instance, there are cases where there are crowd cheer for long periods of time, and there can be multiple scenes between the audio peak and the target point where the crowd is still cheering. Therefore, there is no way to be certain about which camera scene corresponds to the point, and great results would be occasional. Moreover, there are many scenarios where the crowd cheers, such as before every important point, whenever the hawk eye is called, or if there is an argument between the players and the judge.

In our tests, we found that using audio peaks for point detection was somewhat effective in the two matches that consisted of the final round. However, the accuracy drastically decreased for the women's semifinal, as the crowd cheerings were not as consistent. In a typical match, the fans usually cheer the most in the important points, and are usually biased towards one player for varying reasons such as being the local player or the underdog player. In unimportant points, the cheerings after the points are not higher than the cheerings before the points.

Using the audio peaks was not as effective in detecting points on its own. However, given the list of camera scenes detected with computer vision techniques, it can filter out false positives and recognize the points from both players. After using the audio peaks of the camera shots obtained by court detection, it was possible both to reduce the false positives to 0, while also discarding redundant points such as double faults and unforced return errors.

The benefits in efficiency of the computer vision techniques clearly surpass the trade-off in time consumption.

7.2 FINAL ALGORITHM AND POINT SELECTION

The final algorithm takes three inputs: the final list of point intervals for rallies and aces, the output of the audio analysis portion with the audio peaks, and the output of the sentiment analysis section containing the points where positive sentiment was detected. The idea is to combine the data of the three sections in order to choose the most important points of the match.

The list of point intervals give very useful information about the length of the points. The shortest intervals would correspond to aces, whereas the longest to rallies. In our approach, we only consider the rallies. We decided not to consider aces from this technique, even though they are potentially useful for future work.

The objective of the final algorithm is to select the desired amount n of tennis points from the match. In this study, n was chosen to be 20. However, it should be noted that it is possible to generate very compact highlights and extended highlights by changing this parameter. The final list of points should include points that were detected from all of the three techniques. The idea is to include a third of the points from each technique in the final list, and the logic to do so is described next.

The first step is to include the best set of points of length $\frac{n}{3}$ from the sentiment analysis technique. If the length of the list is less than or equal to $\frac{n}{3}$, all the points are appended to the final list of points. However, if the list is large, it is necessary to discard some points, keeping only $\frac{n}{3}$. Because our sentiment analyzer only classifies points as either positive or negative/neutral, we do not have scores that could suggest which are the best among the list. Therefore, the approach taken is to remove points that are the most close to each other, including the points that are as

spaced out as possible. The reason for this is to avoid showing too many points from the same part of the match, as the objective of highlights is to show the relevant content of every portion of it.

After including the points from sentiment analysis, an important step is to remove those points from the lists of best points of the other two techniques. The objective is avoid situations where there is a lot of overlap between the three lists, which can result in the final list of points to be considerable shorter than the desired n .

The inputs of audio wave analysis and rallies have the convenience of being sorted from best to least best. Therefore, the first step would be to slice the list from the beginning to $\frac{n}{3}$ (or $\frac{n - \text{len}(\text{bestSentiment})}{2}$) and check if they should be included. Starting with the list of rallies, it is necessary to do the same check as with sentiment analysis to see if there were more detections than the target number. The approach for selecting spaced out points needs to be different to the one before due to the list being sorted. Here, the points from the beginning should be prioritized. The best way to achieve this behaviour is starting with the left portion of the list of size $\frac{n}{3}$, iteratively removing contiguous points and growing the list to the right.

The points from audio wave analysis are selected with same approach as above. It should be noted that the points added to the final list from computer vision are also removed from the input list of audio analysis to avoid overlap.

Figure 7.1 shows a diagram of the software pipeline, depicting the flow of the data step by step going through the different techniques in CV, NLP and AA. As the diagram suggests, the software receives a full tennis match with no cuts as its input, applies the techniques and outputs a video of the highlights.

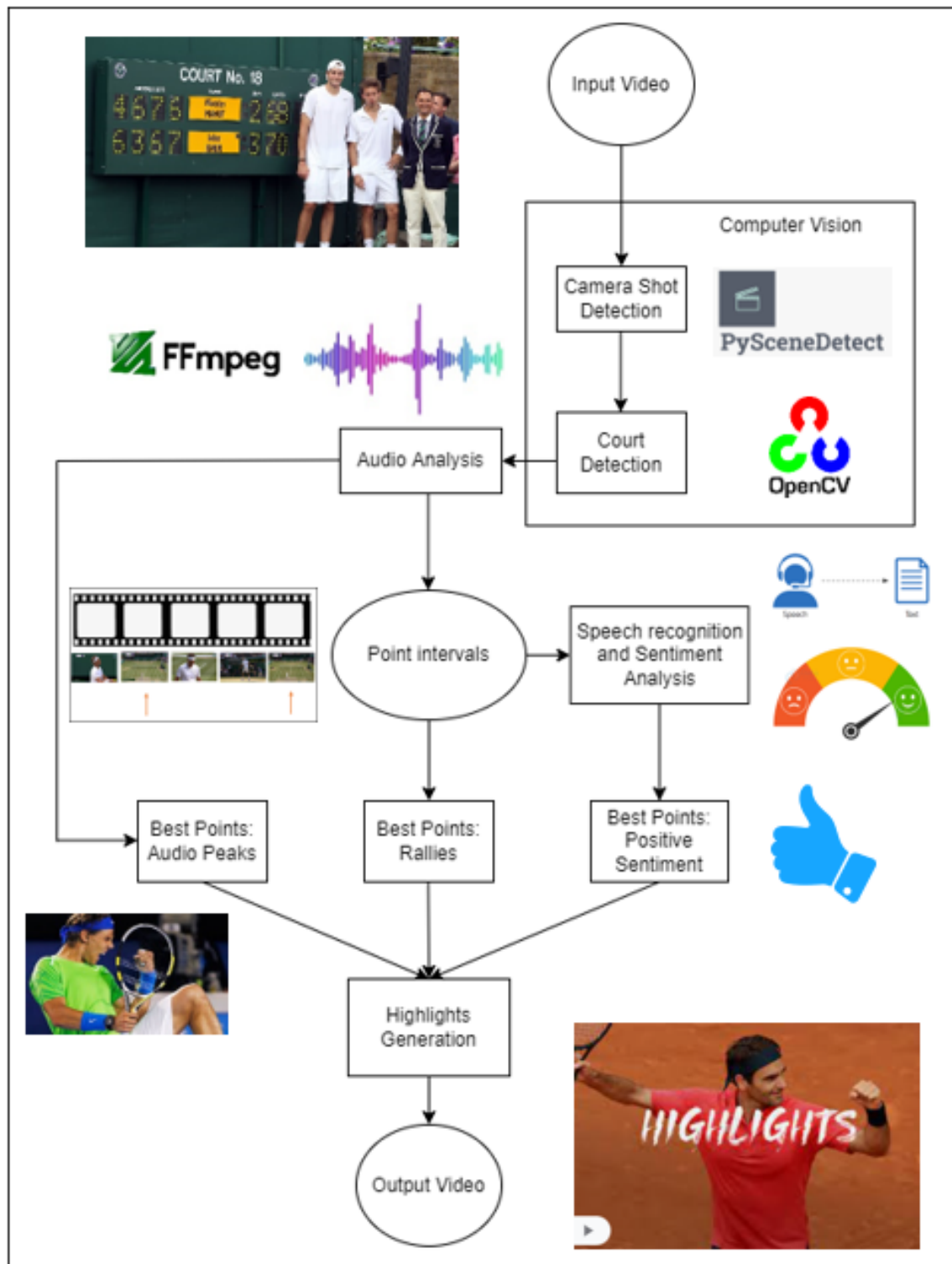


Figure 7.1: Software Pipeline

7.3 HIGHLIGHTS GENERATION FOR FULL TENNIS MATCHES

The software was tested using the three full matches of the US Open 2021 to create highlights of 20 points. The men's final and women's semi final consist of three sets each, and have a duration of 1 hour and 55 minutes. The women's final consists of 2 sets, and is 1 hour and 39 minutes long. These matches were chosen as their length were the closest found to the average tennis match of an hour and 30 minutes. The great majority of the available matches on YouTube are exceptionally long, with 5 sets lasting over 5 hours.

Table 7.4: 2021 US Open men's final

Task	Current Camera Shots	Execution Time
PySceneDetect	963	13m
Players and Court Detection	227	31m
Audio Analysis	-	1m
Redundant Points Removal	148	0
Speech Recognition	-	6m
Sentiment Analysis	-	0
Points Selection	20	0
Highlights Generation	-	23m

Table 7.5: 2021 US Open women's final

Task	Current Camera Shots	Execution Time
PySceneDetect	798	8m
Court Detection	188	23m
Audio Analysis	-	1m
Redundant Points Removal	130	0
Speech Recognition	-	6m
Sentiment Analysis	-	0
Points Selection	20	0
Highlights Generation	-	33m

Table 7.6: 2021 US Open women's semifinal

Task	Current Camera Shots	Execution Time
PySceneDetect	990	14m
Court Detection	253	27m
Audio Analysis	-	1m
Redundant Points Removal	153	0
Speech Recognition	-	7m
Sentiment Analysis	-	0
Points Selection	20	0
Highlights Generation	-	28m

7.4 EXECUTION TIME

Tables 7.4, 7.5 and 7.6 show the results when inputting the full videos to the software at each phase. In terms of execution time, the entire process to generate highlights took around 1 hour and 15 minutes to complete for the three matches. Although this is considerably time consuming, the models were definitely optimized as much as possible to avoid analyzing unnecessary frames. Moreover, this result could be improved if a more powerful hardware was available. Google Colaboratory pro offers the GPUs NVIDIA Tesla K80, T4, and P100. It should be noted that the execution time of our software is shorter than creating highlights manually, which would involve watching the entire match and deciding which points are best.

Besides court detection, the phases that were time consuming were shot detection with PySceneDetect and the generation the output video with FFMPEG, which ranged from 8 to 14 and 23 to 33 minutes respectively. Lastly, audio and sentiment analysis executed almost instantly. The audio analysis step only works with audio data, and is therefore less time consuming than the computer vision models that are ran over hundreds of image frames. Sentiment analysis also executes very fast, where the only consuming task is calling the API for speech recognition. The time consumption of the sentiment analyzer is negligible as it only consists of iterating

over a list of sentences and searching for the target words from the two dictionaries of nouns and adjectives.

Apart from Google Colaboratory Pro and Google Drive One, the only other paid service in the software pipeline is the Google speech-to-text API to generate the transcripts of the commentators. The speech recognition API billed 15 seconds per point analyzed, which corresponds to 2220, 1950 and 2295 seconds in total for men's final, women's final and semifinal. By being able to analyze only specific time intervals, it was possible to substantially reduce the amount of time for the API to process. Conversely, in the absence of the lists of points, the whole video would need to be sent to the API, which is not feasible in terms of costs.

7.5 POINT SELECTION

As discussed beforehand, the detection model for the tennis court is very efficient in reducing the number of camera shots to consider. After applying shot detection with PySceneDetect, the total numbers of camera shots detected were 963, 798 and 990 for the three matches. In these tests, court detection performed was used to extract 227, 188 and 253 camera shots from the entire list by identifying the points in play. Later on, audio analysis was performed to remove the redundant points, bringing these number down to 148, 130 and 153 shots that correspond to points. Finally, the lists of best points from sentiment analysis, audio wave analysis and rallies was inputted to the final algorithm, which selected the best 20 points for the output highlights.

Tables 7.7, 7.8 and 7.9 shows the best points that were detected by each of the three techniques for the three matches. Each number represents a point from the

Table 7.7: Point Selection for 2021 US Open men's final

Technique	Points
Rallies	81, 69, 106, 63, 59, 7, 146, 129, 145, 52, 29, 87
Sentiment Analysis	3, 5, 6, 13, 19, 44, 47, 66, 87, 107, 110, 126, 127, 133
Audio Analysis	139, 50, 140, 144, 73, 142, 64, 101, 63, 80, 102, 127, 126, 147, 146, 69, 71, 109, 106, 148
Final List	3, 7, 44, 50, 59, 63, 64, 66, 69, 73, 80, 81, 87, 101, 106, 107, 126, 127, 139, 142

Table 7.8: Point Selection for 2021 US Open women's final

Technique	Points
Rallies	127, 2, 50, 121, 54, 64, 119, 49, 67, 116, 114, 31, 24
Sentiment Analysis	1, 17, 19, 32, 39, 41, 53, 55, 60, 97, 106, 128
Audio Analysis	107, 76, 120, 125, 66, 31, 9, 86, 93, 121, 127, 114, 119, 130, 50, 49, 124, 16, 54, 109
Final List	1, 2, 17, 31, 32, 49, 53, 54, 64, 66, 67, 76, 97, 107, 116, 119, 120, 125, 127, 128

Table 7.9: Point Selection for 2021 US Open women's semifinal

Technique	Points
Rallies	91, 3, 148, 109, 47, 107, 21, 20, 141, 8, 105, 23, 54, 13
Sentiment Analysis	7, 27, 30, 49, 52, 57, 58, 65, 68, 72, 73, 83, 114, 130, 138, 140
Audio Analysis	60, 134, 137, 92, 58, 50, 47, 78, 91, 21, 119, 84, 30, 126, 83, 152, 142, 148, 19, 76
Final List	3, 7, 20, 21, 30, 47, 50, 52, 58, 78, 83, 84, 91, 92, 107, 114, 119, 134, 137, 138

ordered list of points from each match. The approach discussed in Chapter 7.2 is performed in order to equally select points from the three techniques, handling the issues of contiguous points and overlaps. The final list in the tables shows how the number of points is equally chosen from the techniques. It can also be seen that the points selected are spaced out along the match, getting the important points of every part of it.

Due to the subjective nature of this task, the efficiency of the point selection had to be manually validated. With our experience in this sport, we were able to confirm

that the points selected from each of the techniques actually corresponded to the important sections of the match, and the outputted highlights resulted in a very descriptive and appealing summaries of the three 2021 US open tennis matches.

For the three highlights generated, the quality of the points from each technique was reviewed. We found the three techniques were able to extract high quality points to include in the highlights. The highlights generated by our software contain points that excite the crowd the most, either due to an impressive quality of tennis or the importance of the point. Long rallies included depicted some of the most notable points in the match, which might not be cheered enough by the crowd. Even though rallies not necessarily generate a lot of emotion in moments that are not very tense and decisive such as the beginning of a game, these points are very valuable for highlights and should be included. With sentiment analysis it was possible to extract some points that were highlighted by the commentators. These points are important because are representative of the best level of tennis that the players achieve in a match.

The overall performance of the techniques and the quality of the highlights met our expectations. However, in our review of the points we found a minority of points from each technique that might not be good enough for the highlights. Firstly, for sentiment analysis, we found that some of the points were misclassified as positive. We found some cases where the commentators were not focused on the points, and were rather referring about something else. Secondly, in the US open men's final match we found that some rallies were misclassified with a great portion of the camera shot was due to a player missing multiple serves and taking a lot of time between them. Lastly, we found that audio analysis might be unfair if the crowd is biased in a tennis match. In the women's semifinal, the crowd was cheering more towards one of the players, and therefore the favourite player won in the majority of the best points detected by this technique.

7.6 SOFTWARE LIMITATIONS

One limitation in this project was the low quality of the commentators' transcripts achieved by the Google API. Even though this service was the best out of the few that were considered and tested, we could see that there was a considerable amount of wrong words when checking the translations. However, due to the simplicity of our sentiment analysis model, which only looks for specific keywords instead of general context, we were still able to find some of the impressive points.

We also found that although the software obtains a very accurate list of camera shots that correspond to points, there are scenarios where the camera shot does not change when a player serves multiple times due to lets. Even though these cases are uncommon, they can be mistakenly classified as rallies as all the missed serves were contributing to the total length. One way we found to go around the previous problem is to include long rallies only if the audio peak was higher than half of the points in order to discard long intervals that include a lot of serves.

The highlights generated fulfilled our expectations, including the best points from multiple techniques and creating a great summary of the match. However, from our judgement, some important points were missing such as set and match points. The reason is that our approach is not able to recognize which points actually correspond to these moments, as it does not take the score into consideration. In the next chapter, future work for detecting these key points with certainty will be discussed, as well as other ways to improve our project.

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

In the last decades, artificial intelligence has been drastically evolving a wide array of domains both in industry and academia. It has been used to solve specific problems that previously required extensive manual labor. One of the areas that was impacted by the rising research in AI is sports. AI has evolved sports in different areas such as healthcare, performance and customer experience. In this paper, multiple subfields of AI were investigated in order to solve the highlights generation task for tennis matches.

This project successfully leveraged techniques in computer vision, natural language processing and audio analysis to produce highlights given an input video of a full tennis match with no cuts. Professional tournaments are still manually gathering data of the match such as the time intervals of the points, which other studies rely on. Our software fulfilled the initial goal of solving this task without the need for any type of manual help.

One of the biggest challenges was to extract the point intervals from the match. The most effective approach tried used shot detection with PySceneDetect, court detection with hough transform and OpenCV, and then audio analysis to remove false positives. Afterwards, the best points were selected from the longest rallies, highest audio peaks and positive comments from the commentators. We manually validated the quality of the highlights generated to conclude that they are a

descriptive summary of tennis matches, with most the points chosen being the most interesting of the matches tested.

Even though our project successfully met the goals that were initially set, there are many ways in which it can be taken to the next level that were not doable due to constraints in time and resources.

8.1 THREATS TO VALIDITY

One limitation of this project is testing the results and quality best points selected. We were able to test and validate the efficiency of each technique on their respective tasks. However, the selection of the points made by the final algorithm is subjective and the quantification of the performance is based on our own opinion given our experience in the sport.

The appropriate way of testing the quality of our highlights is to conduct an observational study. This would consist of getting a random sample of tennis fans that have watched official highlights in the past and are not related to us in any way. An appropriate observational study would be to show both the generated highlights by the project and the official highlights for the same tennis match, and then fill a survey where they can rate the quality without bias.

8.2 FUTURE WORK

Further research could also be done in order to improve the overall quality of the highlights generated. Firstly, one limitation found was the extra time of point scenes due to missed serves and lets. One way to tackle this issue is to develop a model for detecting the sound of the net, which is very specific when serving due to the

let sensor. Moreover, speech recognition can also help to look for 'out' and 'let' statements of the umpire that indicate missed serves.

Our project could also be improved with access to a better speech-to-text service. It is possible for a speech-to-text service to be good enough to clearly transcribe tennis matches even when there is a lot of background noise. Our transcriptions were compared to the subtitles of the videos in YouTube, where the quality was noticeably superior. More accurate transcripts would not only have an instant positive impact to our results, but could also be leveraged by a more sophisticated sentiment analyzer that requires a better transcript. While our approach only classifies points as positive or neutral/negative, a more complex sentiment analyzer would allow to understand context and assign scores within a range to more accurately describe how positive the comments are. This way, it is possible to sort the points detected by this technique from most to least positive and select the best points.

An additional technique that would considerably improve our project is Optical Character Recognition (OCR). OCR is a computer vision technique that consists of extracting written text from images using neural networks. This technology could be leveraged by our project in order to extract the scoreboard from the screen. In tennis matches, the scoreboard is always placed in the bottom-left corner and is shown whenever points are in play. Other studies have already successfully implemented OCR to solve the scoreboard extraction task for different sports including tennis, soccer, basketball and rugby [32, 27].

The score provides very valuable information about the timeline of the match. Being able to track the score allows to select specific important moments such as break, set and match points with high accuracy. Our current approach is usually able to select those points, however, the OCR technique would provide the certainty that these key points are always included.

This project could be taken to the next level by incorporating an interactive User

Interface (front end), and evolving our software to become a full stack application. This way, the backend would consist of our current software storing the results in a database in real time, such that the front end can fetch this data to create useful visualizations. The idea of this UI is for the user to be able to reproduce the full video and have sliders for each of the techniques representing the timeline of the input video containing ticks with the location of the relevant points detected. Therefore, the user can manually check the quality of the highlights and has the ability to add and remove as many points as desired from each technique. This sliders would be accompanied by relevant information such as a visualization of the graph amplitude vs time for audio analysis, the keywords detected for sentiment analysis, and the type of point based of length (rally or ace). Although implementing this idea would have been too time consuming to be done this year, the benefits could be substantial in terms of user experience.

REFERENCES

1. Khan Academy. Gradient descent. <https://www.khanacademy.org/math/multivariable-calculus/applications-of-multivariable-derivatives/optimizing-multivariable-functions/a/what-is-gradient-descent>. 17
2. Charu C. Aggarwal. *Neural Networks and Deep Learning*. Springer International Publishing, 2019. 8, 9, 10
3. aravindpai. Cnn vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning. <https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/>, 2020. xi, 10
4. ArtLabs. Tennis Tracking. <https://github.com/ArtLabss/tennis-tracking>. xii, 49, 50
5. Anant Baijal, Jaeyoun Cho, Woojung Lee, and Byeong-Seob Ko. Sports highlights generation based on acoustic events detection: A rugby case study. *2015 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2015. doi: 10.1109/icce.2015.7066303. URL <http://dx.doi.org/10.1109/ICCE.2015.7066303>. 27
6. Maxime Bataille. Track tennis players and the ball. https://github.com/MaximeBataille/tennis_tracking. 49
7. Aaron Baughman, Eduardo Morales, Gary Reiss, Nancy Greco, Stephen Hammer, and Shiqiang Wang. Detection of Tennis Events from Acoustic Data. In *Proceedings Proceedings of the 2nd International Workshop on Multimedia Content Analysis in Sports, MMSports '19*, page 91–99, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369114. doi: 10.1145/3347318.3355520. URL <https://doi.org/10.1145/3347318.3355520>.
8. Jake Bouvrie. Notes on Convolutional Neural Networks. November 2006. URL <http://cogprints.org/5869/>. 18
9. Jason Brownlee. A Gentle Introduction to Object Recognition With Deep Learning. <https://machinelearningmastery.com/object-recognition-with-deep-learning/>, 2021. 7

10. Maxim Bruckheimer and Arcavi Abraham. Farey Series and Pick's Area Theorem. *The Mathematical Intelligencer*, 17:64–67, 1995.
11. Ed Burns. What is a neural network? explanation and examples. <https://www.techtarget.com/searchenterpriseai/definition/neural-network>, 2021. 9
12. Brandon Castellano. PySeneDetect, Intelligent scene cut detection and video splitting tool. <https://pyscenedetect.readthedocs.io/en/latest/>. 32
13. Adroz Chakure. YOLOv3 (You Only Look Once) Deep Learning Algorithm. <https://medium.datadriveninvestor.com/yolov3-you-only-look-once-12de76ad74d5>, 2021. xi, 36, 38
14. Google Cloud. Speech-to-Text. <https://cloud.google.com/speech-to-text>.
15. B.J. Copeland. Artificial Intelligence. <https://www.britannica.com/technology/artificial-intelligence>. 1
16. Maria Alejandra Coy Ulloa. Forward and Back Propagation over a CNN. <https://www.linkedin.com/pulse/forward-back-propagation-over-cnn-code-from-scratch-coy-ulloa/>. xi, 18
17. Musab Coşkun, Ayşegül Uçar, Özal yıldırım, and Yakup Demir. Face Recognition Based on Convolutional Neural Network. 11 2017. doi: 10.1109/MEES.2017.8248937. 11, 13
18. Juliana Delua. Supervised vs. Unsupervised Learning: What's the Difference? <https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning>, 2021. 6
19. FFmpeg Developers. FFmpeg. <https://www.ffmpeg.org/ffmpeg.html#Synopsis>, 2021. 51
20. e Echocardiography. Sound Wave Properties. <https://e-echocardiography.com/page/page.php?UID=1429454151>. 25
21. IBM Cloud Education. Convolutional Neural Networks. <https://www.ibm.com/cloud/learn/convolutional-neural-networks>, 2020. xi, 12, 13
22. IBM Cloud Education. Machine Learning. <https://www.ibm.com/cloud/learn/machine-learning>, 2020. 6

23. IBM Cloud Education. Speech Recognition.
<https://www.ibm.com/cloud/learn/speech-recognition>, 2020. 24
24. Factoreal. How Artificial Intelligence is Changing the Sports Industry.
<https://www.factoreal.com/blog/ai-changing-the-sports-industry>. 2
25. Guilherme Fião, Teresa Romão, Nuno Correia, Pedro Centieiro, and A. Eduardo Dias. Automatic Generation of Sport Video Highlights Based on Fan's Emotions and Content. In *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology, ACE '16*, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450347730. doi: 10.1145/3001773.3001802. URL <https://doi.org/10.1145/3001773.3001802>.
26. Morgan R. Frank, David Autor, James E. Bessen, Erik Brynjolfsson, Manuel Cebrian, David J. Deming, Maryann Feldman, Matthew Groh, José Lobo, Esteban Moro, Dashun Wang, Hyejin Youn, and Iyad Rahwan. Toward understanding the impact of artificial intelligence on labor. *Proceedings of the National Academy of Sciences*, 116(14):6531–6539, 2019. ISSN 0027-8424. doi: 10.1073/pnas.1900949116. URL <https://www.pnas.org/content/116/14/6531>. 1
27. Anurag Ghosh and C. V. Jawahar. Smarttennistv: Automatic indexing of tennis videos, 2018. 77
28. Edwin “Jed” Herman Gilbert Strang. Calculus.
<https://openstax.org/books/calculus-volume-3/pages/4-6-directional-derivatives-and-the-gradient>, 2016. 16, 17
29. Ross Girshick. Fast R-CNN, 2015. xi, 19, 20
30. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. xi, 19
31. Onesmus Mbaabu Grace Karimi. Introduction to YOLO Algorithm for Object Detection. <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection>, 2021. xi, 33
32. Jinlin Guo, Cathal Gurrin, Song-Yang Lao, Colum Foley, and Alan Smeaton. Localization and recognition of the scoreboard in sports video based on sift point matching. pages 337–347, 01 2011. ISBN 978-3-642-17828-3. doi: 10.1007/978-3-642-17829-0_32. 77
33. MK Gurucharan. Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network, Apr 2021. URL <https://www.upgrad.com/blog/basic-cnn-architecture/>. xi, 11, 12, 14

34. Hackernoon. Intro to Audio Analysis: Recognizing Sounds Using Machine Learning. <https://hackernoon.com/intro-to-audio-analysis-recognizing-sounds-using-machine-learning-qy2r3ufl>, 2020. xi, 25, 26
35. Alan Hanjalic. Adaptive extraction of highlights from a sport video based on excitement modeling. *Multimedia, IEEE Transactions on*, 7:1114 – 1122, 01 2006. doi: 10.1109/TMM.2005.858397.
36. Software Testing Help. What Is Artificial Intelligence: Definition Sub-Fields Of AI. <https://www.softwaretestinghelp.com/what-is-artificial-intelligence>, 2022.
37. R Horan and M Lavelle. Gradients and Directional Derivatives. https://www.plymouth.ac.uk/uploads/production/document/path/3/3742/PlymouthUniversity_MathsandStats_gradients_and_directional_deriviatives.pdf, 2004.
38. Yo-Ping Huang, Ching-Lin Chiou, and Frode Eika Sandnes. An Intelligent Strategy for the Automatic Detection of Highlights in Tennis Video Recordings. *Expert Syst. Appl.*, 36(6):9907–9918, aug 2009. ISSN 0957-4174. doi: 10.1016/j.eswa.2009.01.078. URL <https://doi.org/10.1016/j.eswa.2009.01.078>. xi, 27, 29, 30
39. Yu-Chuan Huang, I-No Liao, Ching-Hsuan Chen, Tsi-Uí Ik, and Wen-Chih Peng. TrackNet: A Deep Learning Network for Tracking High-speed and Tiny Objects in Sports Applications, 2019. xi, xii, 39, 40, 41, 44
40. IBM. Computer Vision. <https://www.ibm.com/topics/computer-vision>. 5
41. Ali Javed, Khalid Bajwa, Hafiz Malik, and Aun Irtaza. An Efficient Framework for Automatic Highlights Generation from Sports Videos. *IEEE Signal Processing Letters*, 23:1–1, 07 2016. doi: 10.1109/LSP.2016.2573042.
42. Tomasz Kacmajor. Hough Lines Transform Explained. <https://tomaszkacmajor.pl/index.php/2017/06/05/hough-lines-transform-explained>, 2017. xii, 47
43. Prateek Karkare. Video and Audio Highlight Extraction Using Python. <https://medium.com/swlh/video-and-audio-highlight-extraction-using-python-40366ee9302b>, 2020. 51
44. Monkey Learn. Sentiment Analysis: A Definitive Guide. <https://monkeylearn.com/sentiment-analysis>. 24
45. Soet Lee. Lines Detection with Hough Transform. <https://towardsdatascience.com/lines-detection-with-hough-transform-84020b3b1549>, 2018. xii, 45, 46

46. Yang Liu Li Deng. *Deep Learning in Natural Language Processing*. Springer, 2018. 22
47. Bernard Marr. How Wimbledon is Using AI and Machine Learning Now. <https://www.linkedin.com/pulse/how-wimbledon-using-ai-machine-learning-now-bernard-marr>, 2021. 3, 30
48. Guillermo Martinez Arastey. Artificial Intelligence (AI) In Sports. <https://www.sportperformanceanalysis.com/article/artificial-intelligence-ai-in-sports>, 2021. 2
49. Machine Learning Mastery. How to Choose an Activation Function for Deep Learning. <https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/>, 2021. 15
50. Michele Merler, Khoi-Nguyen C. Mac, Dhiraj Joshi, Quoc-Bao Nguyen, Stephen Hammer, John Kent, Jinjun Xiong, Minh N. Do, John R. Smith, and Rogerio Schmidt Feris. Automatic Curation of Sports Highlights Using Multimodal Excitement Features. *IEEE Transactions on Multimedia*, 21(5): 1147–1160, 2019. doi: 10.1109/TMM.2018.2876046.
51. Michele Merler, Khoi-Nguyen C. Mac, Dhiraj Joshi, Quoc-Bao Nguyen, Stephen Hammer, John Kent, Jinjun Xiong, Minh N. Do, John R. Smith, and Rogerio Schmidt Feris. Automatic Curation of Sports Highlights Using Multimodal Excitement Features. *IEEE Transactions on Multimedia*, 21(5): 1147–1160, 2019. doi: 10.1109/TMM.2018.2876046. 27, 28, 31, 51
52. Umberto Michelucci. *Advanced Applied Deep Learning*. Apress, 2019. xi, 7, 12, 35
53. Patrick Moorhead. Tennis Is Now On The Cloud And Powered By Applied AI, And This Is How Roland-Garros Did It. <https://www.forbes.com/sites/patrickmoorhead/2021/08/03/tennis-is-now-on-the-cloud-and-powered-by-applied-ai-and-this-is-how-roland-garros-did-it/?sh=6e1bec9e73f3>. 3, 30
54. Priyanka Mukhopadhyay and Bidyut B. Chaudhuri. A survey of Hough Transform. *Pattern Recognition*, 48(3):993–1010, 2015. ISSN 0031-3203. doi: <https://doi.org/10.1016/j.patcog.2014.08.027>. URL <https://www.sciencedirect.com/science/article/pii/S0031320314003446>. 45
55. Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning Deconvolution Network for Semantic Segmentation, 2015. xii, 41, 43
56. Roger Hinrichs Paul Peter Urone. *Physics*. OpenStax, 2020. xii, 52, 53
57. Douglas Repetto Mary Roberts Dan Rockmore Phil Burk, Larry Polansky. Chapter 1: The Digital Representation of Sound, Part One: Sound and Timbre. http://musicandcomputersbook.com/chapter1/01_01.php. xi, 26

58. Babette M Pluim. The evolution and impact of science in tennis: eight advances for performance and health. *British Journal of Sports Medicine*, 48(Suppl 1):i3–i5, 2014. ISSN 0306-3674. doi: 10.1136/bjsports-2014-093434. URL https://bjsm.bmj.com/content/48/Suppl_1/i3. 3
59. NLTK Project. Source code for nltk.sentiment.vader. https://www.nltk.org/_modules/nltk/sentiment/vader.html. 58
60. PyTorch. TORCHVISION.MODELS. <https://pytorch.org/vision/stable/models.html#torchvision-models>.
61. Joseph Redmon and Ali Farhadi. YOLOv3: An Incremental Improvement, 2018. 36
62. Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection, 2016. xi, 33, 35
63. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, 2016. xi, 19, 21
64. Ankit Sachan. Detailed Guide to Understand and Implement ResNets. <https://cv-tricks.com/keras/understand-implement-resnets/>.
65. Claire Sanford. Introduction to Speech Recognition Algorithms: Learn How It Has Evolved. <https://www.rev.com/blog/introduction-to-speech-recognition-algorithms>, 2021. 25
66. Jennifer Shalamanov. Why AI Will Replace Some Jobs and Others Will Stick Around. <https://www.udacity.com/blog/2021/02/why-ai-will-replace-some-jobs-and-others-will-stick-around.html>, 2021. 1, 2
67. Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image recognition, 2015. 41
68. Aishwarya Singh. Demystifying the Mathematics Behind Convolutional Neural Networks (CNNs). <https://www.analyticsvidhya.com/blog/2020/02/mathematics-behind-convolutional-neural-network/>, 2020. 19
69. S.W. Smith. *The Scientist and Engineer’s Guide to Digital Signal Processing*. California Technical Pub., 1999. ISBN 9780966017649. URL <https://books.google.com/books?id=ZtfaNwAACAAJ>. 52
70. Jasper Uijlings, K. Sande, T. Gevers, and A.W.M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, 09 2013. doi: 10.1007/s11263-013-0620-5. xi, 20

71. SOLIDWORKS UK. The Evolution of Tennis: How Technology is Advancing the Game. <https://blogs.solidworks.com/solidworksblog/2018/07/the-evolution-of-tennis-how-technology-is-advancing-the-game.html>, 2018. 3
72. Muneeb ul Hassan. Vgg16 – Convolutional Network for Classification and Detection. <https://www.section.io/engineering-education/introduction-to-yolo-algorithm-for-object-detection>, 2018. xii, 42
73. Rachel Wolff. Quick Introduction to Sentiment Analysis. <https://towardsdatascience.com/quick-introduction-to-sentiment-analysis-74bd3dfb536c>, 2020. 23, 58
74. Thomas Wood. Softmax function. <https://deeptai.org/machine-learning-glossary-and-terms/softmax-layer>. 16
75. XueFei Zhou. Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation. *Journal of Physics: Conference Series*, 2018. doi: <https://doi.org/10.1088/1742-6596/1004/1/012028>. xi, 17, 18

