Estimating Performance Criteria in Soccer Using a Passing Machine and Machine Learning Based Video Processing

Domagoj Pavić, Prof. Bernhard Hollaus, PhD (supervisor)

***Abstract*—This paper presents a program that rates the quality of a football(soccer) player's touch, contact with any permissible body part, in football using machine learning and real-time object detection and image segmentation model *YOLOv8*. The goal is to detect touches form a video and count the number of touches as well as assess their effectiveness. The final outcome then reports on how well the ball was received and how impactful it was on the end result being hitting the target goal. The idea is to use *YOLOv8*, developed by *Ultralytics*, to detect the pose of a person and objects(sports balls). Labeling of each touch(frame of touch) is done manually by a custom *Python* script while feature extraction uses the labeled frames and *YOLOv8* is used to determine the relationships between a person and sports balls. Another custom *Python* script is used to extract the data and populate a matrix that is going to be used as a data frame for machine learning. The last step is a sequential neural network that, given the large enough dataset, would predict the frame at which the touch happens and how many there are in a single video session. The neural network uses an LSTM(Long-short term Memory) approach in order to have each frame depend on the previous ones and not work independently.**

***Index Terms*—YOLOv8, Python, Football(soccer), Machine Learning, LSTM, Neural Network**

I. Introduction

In recent years, technology has profoundly transformed various aspects of football (soccer), from officiating to performance analysis. One of the most significant advancements lies in the domain of data analytics, where video recordings and sophisticated algorithms have become indispensable tools for enhancing player performance. Moreover, to have a chance of evaluating performance there must first be a touch recognized by an algorithm. For this purpose *YOLOv8*, a real-time object detection model, will be used for feature extraction frame by frame. Using *YOLOv8* for object detection and pose estimation a dataset is going to be created upon which the custom algorithm will be trained.

The idea of predicting a touch using solely an algorithm is encapsulated thus this paper in its continuation will explore how to measure the efficiency of a player's touch and if the touch even happened. The main focus is on the recognition of touch as the quality of each is subjective and subjugated to preference and would require further cooperation with Football professionals. Here *YOLOv8* is used because of its ability to reliably detect objects and pose estimations. This is the first step to any and all analysis as without a recognition of touch there is no further processing of data.

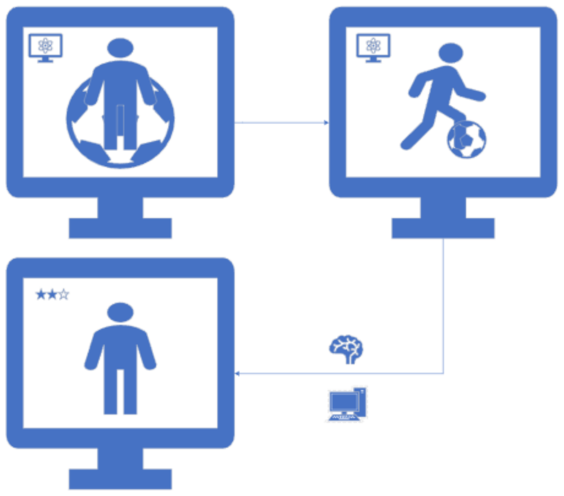


Fig. 1: Big picture

II. Previous work

Technology in football in the last decade has been in a steady rise with the introduction of VAR(Visual assistant referee). VAR uses numerous cameras set up on the field to profile a player and help determine edge cases or rectify mistakes in football 2. The most recent addition is a sensor inside the ball that helps with the determination of the exact time a touch happened 3. On the other hand, this paper focuses on object detection and pose estimation using *YOLOv8* in real time thus requiring less setup once the model is trained 4.

Other than VAR and referee assistance, player analysis is another key focus of industry leading technology. During the match, algorithms are used to collect data on each player individually. In many cases, coaches of those teams use the data to evaluate each player's performance instead of only the feel of how a particular player performed. One such tool is *Opta* (figure 5). *Opta* is a fusion of an expert data analyst and an AI (artificial intelligence) working in parallel to deliver player and team statistics. Machine learning algorithms are used to assess and predict the likelihood of a successful pass, shot, elevation of pressure, etc. This leads to a better understanding of how well a player performs as well as

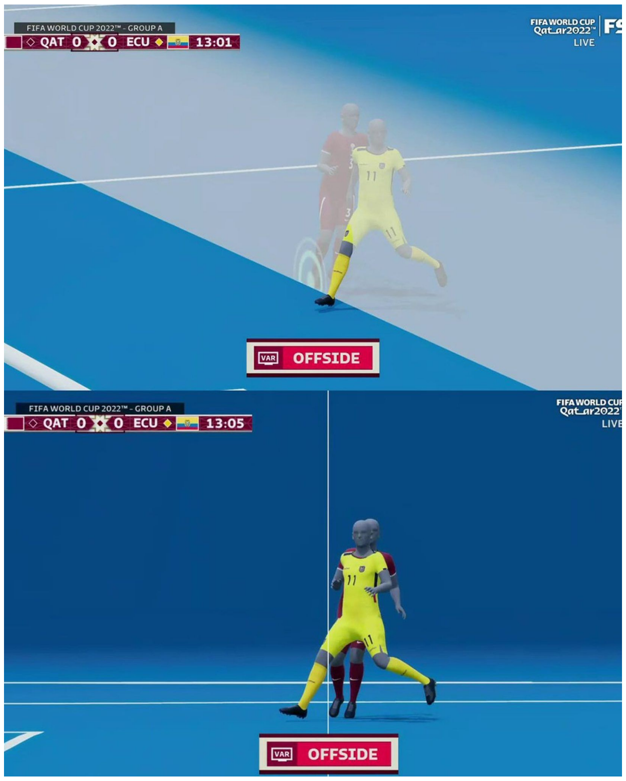


Fig. 2: Semi automated VAR system [1]

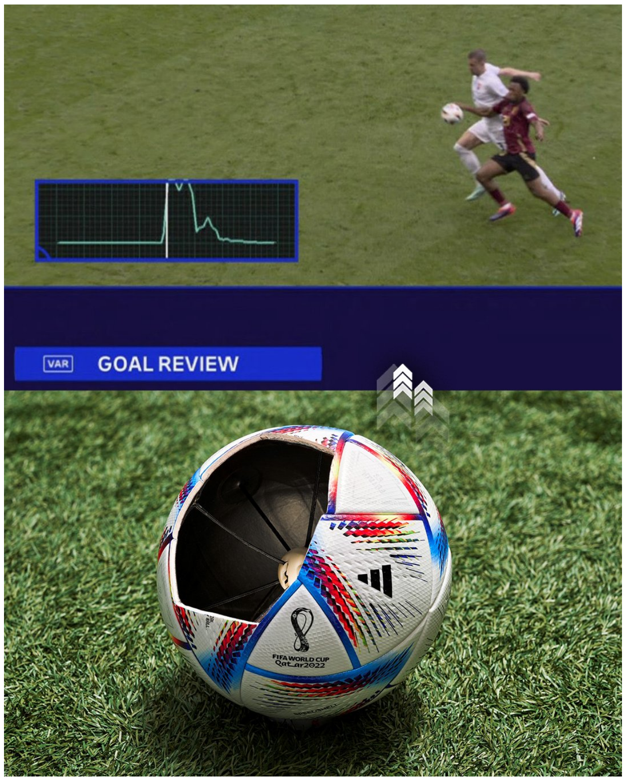


Fig. 3: IMU ball sensor [2]

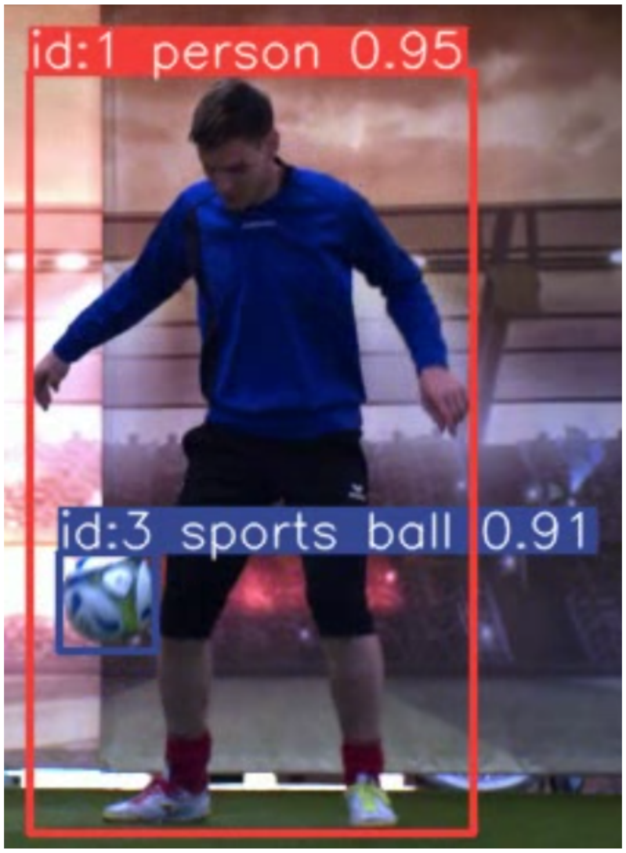


Fig. 4: YOLOv8 object detection.

providing an insight into the effectiveness of each player, their tendencies and play patterns.

Fig. 5: Opta, data collection and analysis company [5].

III. Objectives

The objective of the project is to create a working program that has the ability to predict touches based on data provided by *YOLOv8* ran on each frame. The program is a deep neural network that contains an LSTM(Long-short-term memory) layer at its core with the idea of concatenating frames of a single video so that the algorithm is forced to take into consideration all the frames when classifying. To accomplish this provided data is labeled and features are engineered and extracted from provided videos after which the neural network is trained.

IV. Methods

A. Data Acquisition

For the purpose of data acquisition, a sports gym was used. The gym is equipped with 4 sports cannons that shoot out balls at various speeds, heights, and rotations. With every ball shoot out of the cannon a goal frame lights up and the player is expected to receive the ball and make a placement shoot into the illuminated goal. A camera is placed at a specific spot such that it has an overview of the player and the receiving ball as shown in figure 6. It doesn't have a view of all of the goals as the main idea of this placement is to collect the data of the player receiving the ball as shown in figure 7 and all subsequent touches. The camera has a frame rate of sixty frames per second with each video being three to four seconds long. That means that the majority of the videos have either two hundred and forty or two hundred and nine frames as will be apparent in later chapters. All of the video acquisition was conducted before this project as a stepping stone for further analysis.

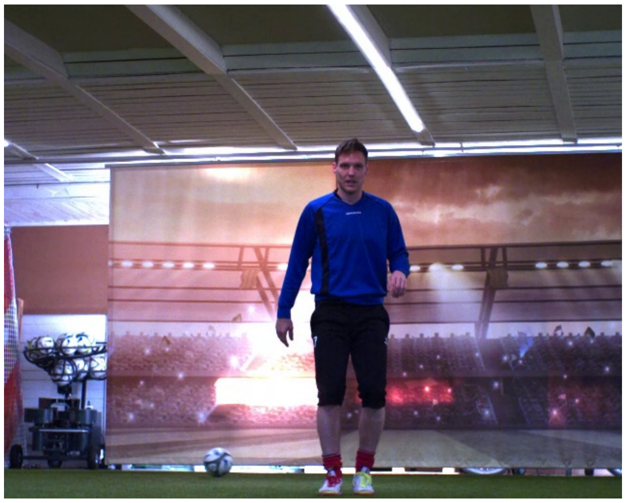


Fig. 6: Player in stand by for the incoming ball.

B. Data Labeling

Provided videos are labeled using a custom *Python* script. In order for the labeler to have a UI (user interface), *PyQt6*, a *Python* library, is used. *PyQt6* provides a plethora of widgets, sliders, buttons and labels 8. The idea is to use the library to loop through a folder of videos labeling each one with ease and storing all the possible classes, total frame count of the video, frames at which a class button was pressed (presumably when a user deemed a touch happened) and time in milliseconds of the frame at which a class button was pressed. Information gained in the previous step is stored into a *.json* file for further processing.

In the current version the *Python* script with the aforementioned library creates two windows. The first window (9) is

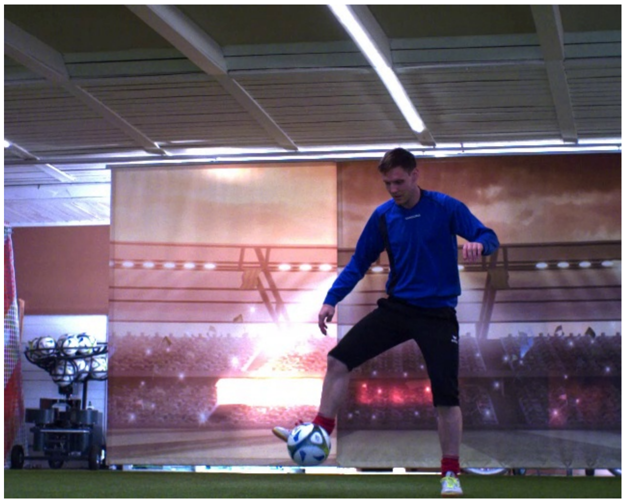


Fig. 7: Player receiving the ball.

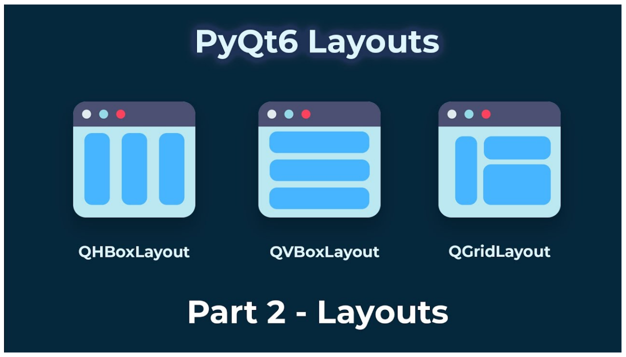


Fig. 8: PyQt6 Layout [3]

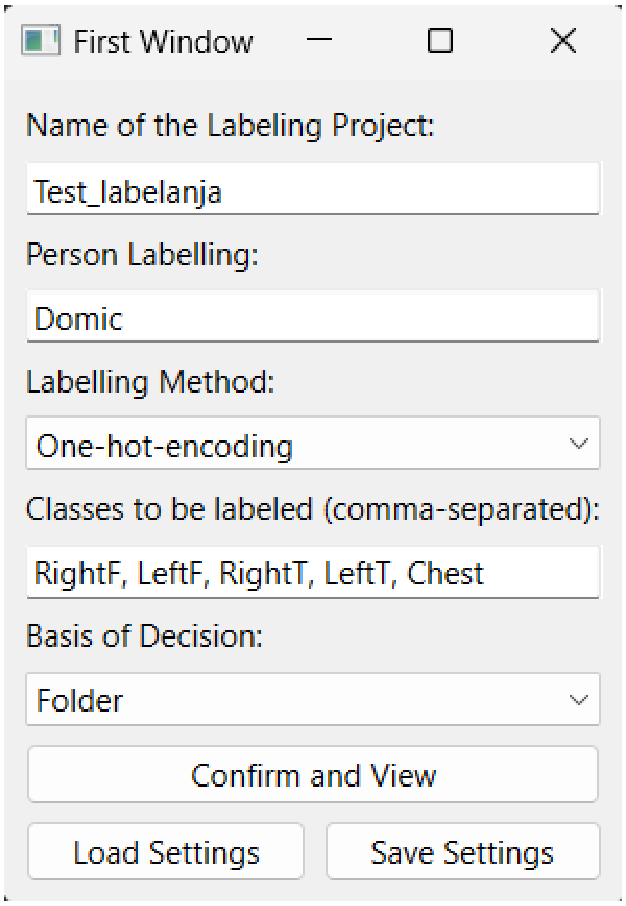
used to give the labeling project a namespace, labeling method (preparation for classification method), desired classes and a decision basis (is the file loaded going to be a folder, video or audio). This window also allows the user to save the configuration for reuse.

The next step is the actual labeling window that has all the information inputted in the first window at the top, a box for showcasing the video, a button for selecting a folder with videos and a folder to save the *.json* files in, button for loading the next video (saves the current data as a *.json* file, start/stop button, slow motion button, video length slider as well as all the class buttons. This is represented in figure 10.

For a clear user experience a section is reserved for a color and numerical representation of the pressed class and its corresponding frame as shown in figure 11.

C. YOLOv8

YOLOv8 is a real-time object detection and image segmentation mode developed by *Ultralytics*. Developers claim unparalleled performance in terms of speed and accuracy as one can attest on their webpage [4]. Some advantages of using *YOLOv8* are: real-time detection, high accuracy, efficient processing and robustness to variations. Real-time detection enables processing of videos but in the scope of this work


First Window

Name of the Labeling Project: Test_labelanja

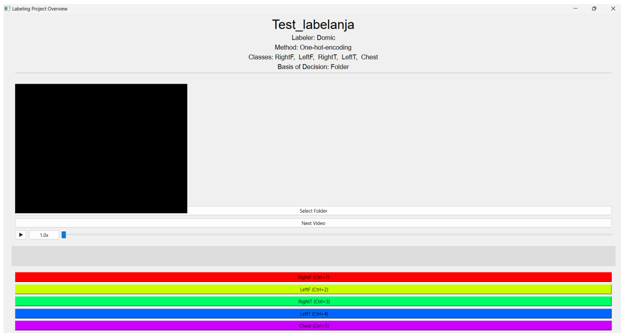
Person Labelling: Domic

Labelling Method: One-hot-encoding

Classes to be labeled (comma-separated): RightF, LeftF, RightT, LeftT, Chest

Basis of Decision: Folder

Fig. 9: First window from the labeling code


Labeling Project Overview

Test_labelanja

Labeler: Domic

Method: One-hot-encoding

Classes: RightF, LeftF, RightT, LeftT, Chest

Basis of Decision: Folder

Fig. 10: Second window from the labeling code

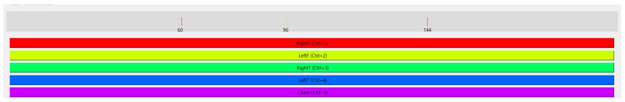


Fig. 11: Label section

YOLOv8 was used on a frame by frame basis, this will be expanded on in chapter !!!!(citation)!!!!. High accuracy is very desirable because it is essential that the ball and player are recognized in consecutive frames and that there are only a few frames that have no or partial object detection. Efficient processing allows the user to switch devices from GPU (graphics processing unit) to CPU (central processing unit) depending on the use case. This allows for an on site analysis as there are no requirements for extensive computational resources. In the context of this project this means that a personal computer with mid specifications is powerful enough to run inference on any data size (nano, small, medium, large and extra large) that *YOLOv8* offers. The data upon which the models in *YOLOv8* are trained is vast and flexible thus enabling detection in different scenarios and environments. This is significant for object detection of a sports ball as there are many variations in patterns as well as *rouge* frames that have less than ideal resolution of a ball (misshaped, malformed, etc.) as shown in figure 12.

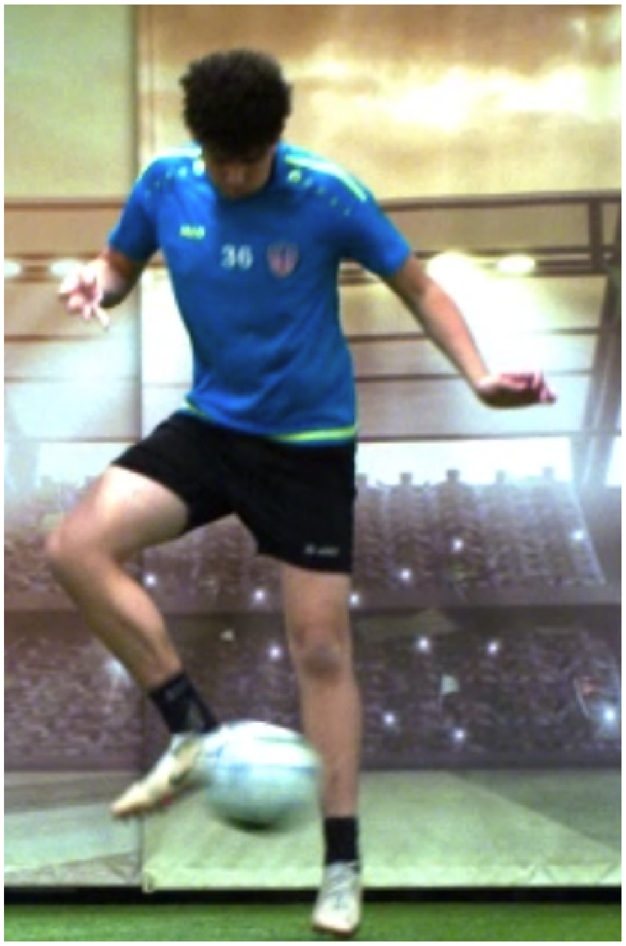


Fig. 12: Deformation of the ball in a frame

V. Feature engineering

With *YOLOv8* data extraction features are created that can encapsulate the needed information for an algorithm to have the best odds of correctly picking up on the classification of a touch and with which body part the touch happened. Since the main focus of this project are object detection and pose estimation information gathered from these two *YOLOv8* inferences are derived from bounding boxes of a ball and a person (as shown in figure 13) and key points (as shown in figure 14) of joints stemming from pose estimation.

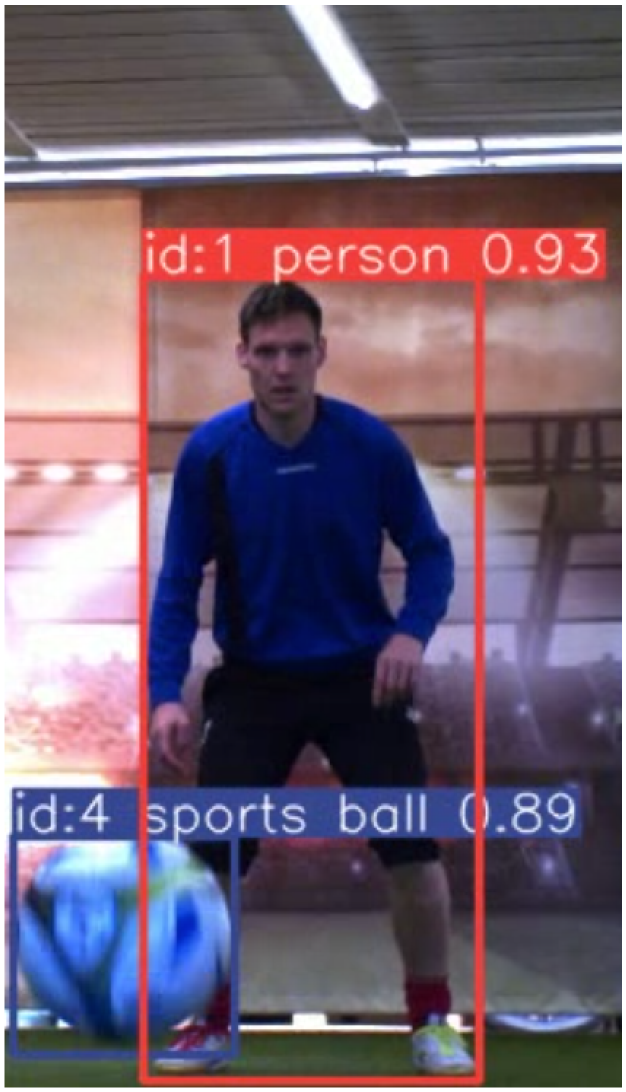


Fig. 13: Object detection bounding boxes.

VI. Dataset Creation

A. Pandas data frame

In machine learning data given into the system is usually a matrix with features and outputs as columns and a set of data as rows. For this reason *Pandas* is used, a popular data frame that is focused on machine learning capabilities and row and column manipulation as well as accessing specific data inside the data frame. Every frame of every video is stored and a pandas data frame is saved as a *.csv* for further post processing before feeding it into a neural network.

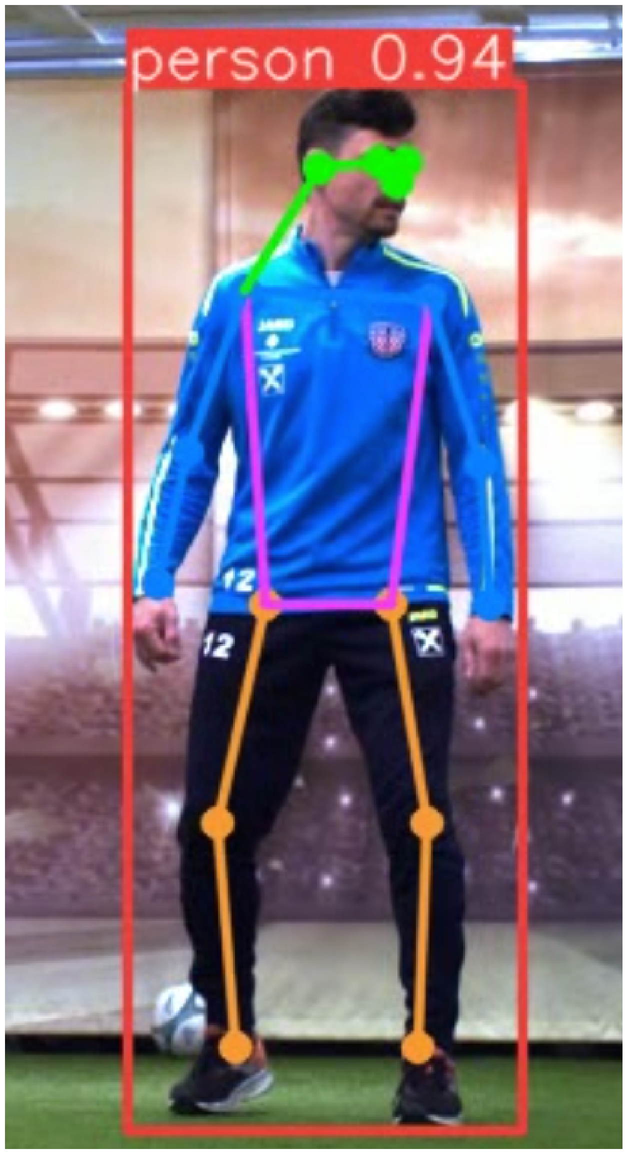


Fig. 14: Pose estimation key points.

B. Feature extraction

There are several obstacles that have to be overcome in order to successfully extract useful features from the frames using *YOLOv8*. Firstly there are cases where two or more people are present in the first frame as shown in figure 15. Secondly, there is the issue of determining the active ball if there are multiple (as shown in figure 16). Finally, there arises the issue of ball tracking. Some frames are lost to object detection and for that reason, the ball object tracking might get interrupted and the active ball lost.

C. Matrix Creation

Creating a matrix means preparing a dataset for an algorithm to train on. Previously mentioned *Pandas* data frame is used for easier manipulation of the data. The columns are named *video_id*, *frame_no*, *distance_to_RF*, *distance_to_LF*,

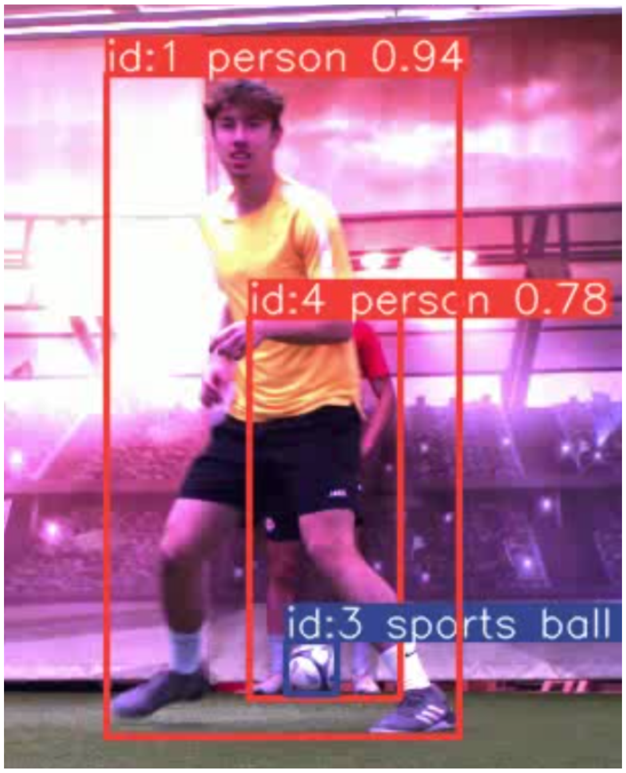


Fig. 15: Multiple people present in the first frame.

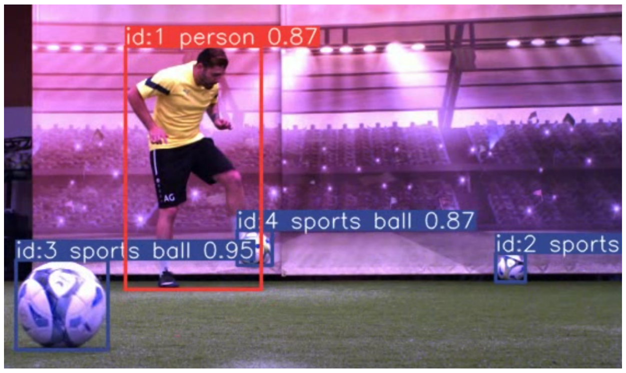


Fig. 16: Multiple balls present in the frame.

distance_to_RT, *distance_to_LT*, *distance_to_CH*, *person_ball_H_rt* for the feature rows, and *RightF*, *LeftF*, *RightT*, *LeftT*, *Chest*, *Other* for one-hot-encoded outputs.

VII. Machine Learning

The first approach is to have a linear (dense) layer expand the number of features through feature combination, then reduce those features with another linear layer. The next is a convolutional one dimensional layer that serves to conduct a further reduction of features. Those features are an input into an LSTM (Long-short-term memory) layer that reduces them even further. The last few steps are a feature expansion using a linear layer and feature reduction with two more linear layers down to the expected size of the output. All of the layers have a rectified linear unit(ReLU) activation function except the last what that has a sigmoid activation function.

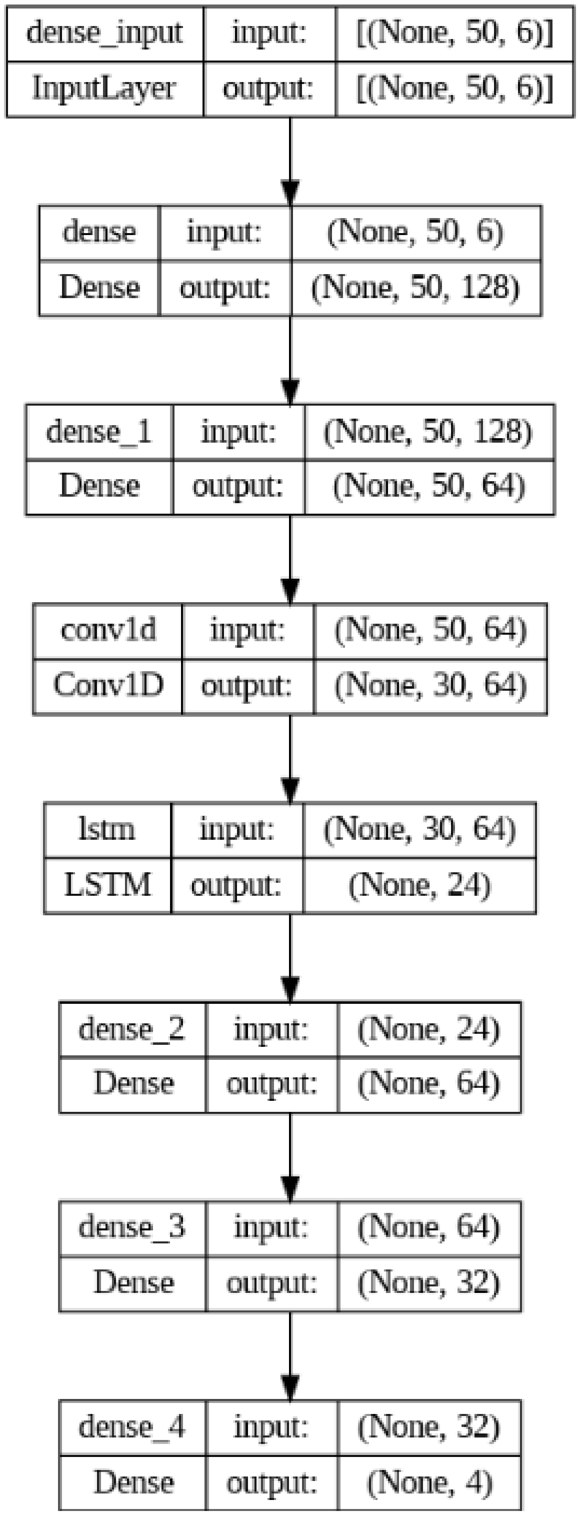

dense_input | input: | [(None, 50, 6)]
InputLayer | output: | [(None, 50, 6)]
dense | input: | (None, 50, 6)
Dense | output: | (None, 50, 128)
dense_1 | input: | (None, 50, 128)
Dense | output: | (None, 50, 64)
conv1d | input: | (None, 50, 64)
Conv1D | output: | (None, 30, 64)
lstm | input: | (None, 30, 64)
LSTM | output: | (None, 24)
dense_2 | input: | (None, 24)
Dense | output: | (None, 64)
dense_3 | input: | (None, 64)
Dense | output: | (None, 32)
dense_4 | input: | (None, 32)
Dense | output: | (None, 4)

Fig. 17: Neural network diagram.

A. *LSTM layer*

Long Short term memory is at the core of the neural network. The problem described is a sequential one, meaning that in order for the network to have the best chance at results the data upon which the network is to be trained has to be connected such that the current frame depends on numerous frames before. The LSTM unit is made up of four feedforward neural networks. Each of these neural networks consists of an input layer and an output layer. In each of these neural networks, input neurons are connected to all output neurons. As a result, the LSTM unit has four fully connected layers. Three of the four feedforward neural networks are responsible for selecting information. They are the forget gate, the input gate, and the output gate. These three gates are used to perform the three typical memory management operations: the deletion of information from memory (the forget gate), the insertion of new information in memory (the input gate), and the use of information present in memory (the output gate). The fourth neural network, the candidate memory, is used to create new candidate information to be inserted into the memory. A diagram of network architecture is shown on figure 18.

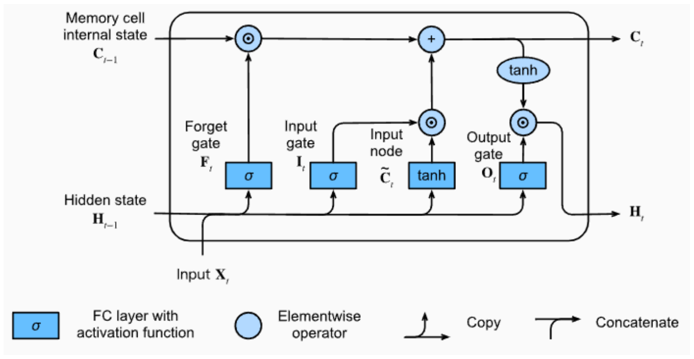


Fig. 18: LSTM network architecture [6].

VIII. Proposed Solution

Firstly the data is distributed to the classes as shown in figure 19. From the bar diagram, it is apparent that one class in particular is dominating. Class *Other* has the most occurrences as every frame that has no touch is automatically labeled as *Other*. A video usually has two or three touches, which means that at most there are twenty one frames labeled as a touch (out of two hundred forty or two hundred nine frames in total). In addition, touch instances are split between all classes, but the class *Other*. This kind of distribution would overfit the algorithm in training to predict only the class *Other*. To try to circumvent this issue the *Other* is split into two different classes, *Other_F* and *Other_NF*. *Other_F* represents a case where the active ball and the player are in the same frame so the features columns are filled with data but no touch has occurred. *Other_NF* on the other hand, is a case where either the active ball, the player, or both are missing from the frame. That means that the data is the default one (820 for all the features except for the ratio which is set to zero). Another issue is that classes *RightT* and *LeftT* are relatively low in occurrences, so those classes are combined together with *RightF* and *LeftF* to create a new class called *Legs*. Finally, class *Chest* is not edited in order to create a distinction between the classes. The distribution in this case is shown in figure 20.

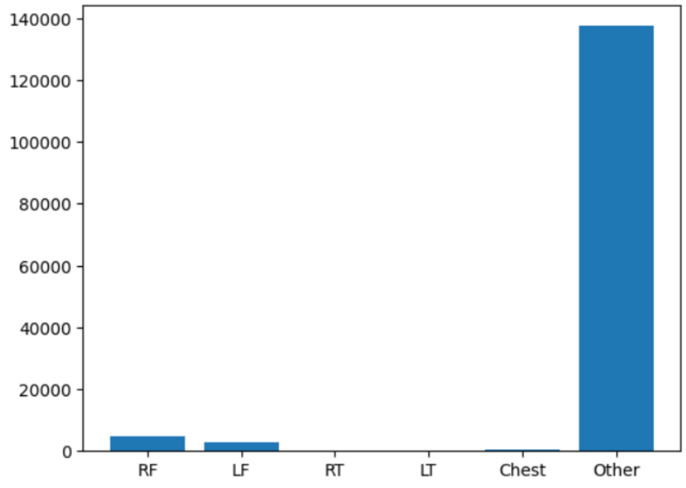


Fig. 19: Class distribution bar diagram .

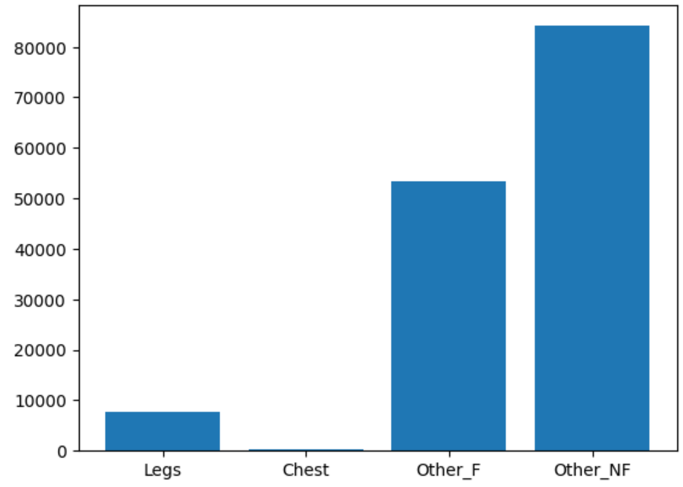


Fig. 20: Class distribution after separation of the data bar diagram .

IX. Results

The original network has only 350 instances out of around 140,000. This is a significant drop in feature count and the results reflect that. Instead, the best results yielded from a different setup, one with three classes. The differences in data are best shown in a bar diagram as in figure 21, where it can be seen that the lowest occurring class *Feet* as a significantly bigger count than the class *Chest*.

The accuracy of this network is reported as 93 percent on a test data set. This is apparent in the confusion matrix shown in figure 22.

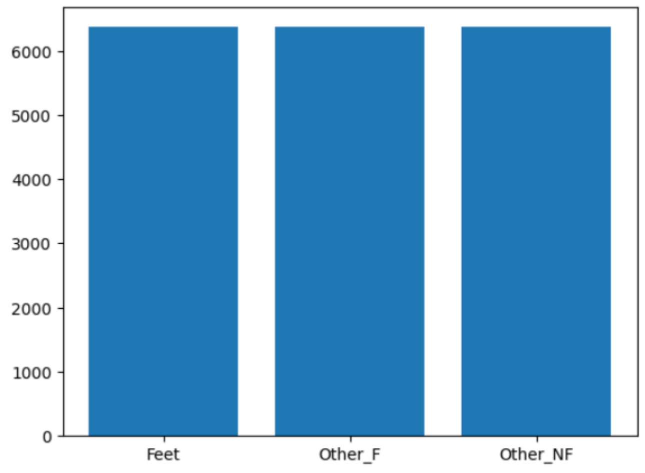


Fig. 21: Bar diagram with only three classes.

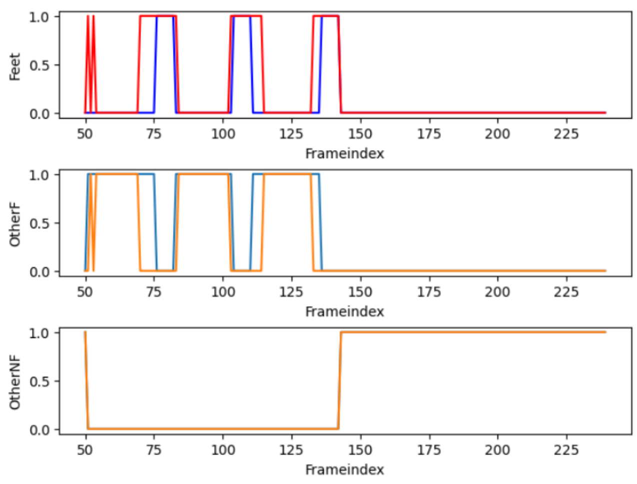


Fig. 23: Performance graph of three classes.

A referral to the prediction graph gives an insight into how accurate does the network perform on a single video taken from the data set at random. Figure 23 represents a prediction graph. Apart from a little offset to the touch predictions and two rouge frames at the start of the measurement the prediction follows the real behaviour. One reason for bigger windows of touch data that comes to mind is in how the data extraction is set up. There is a possibility that some frames before in this case have equally strong claims of being a touch as those labeled ones, as the ball could have moved right over the foot used to make the touch a couple of frames before the touch was labeled and expanded to.

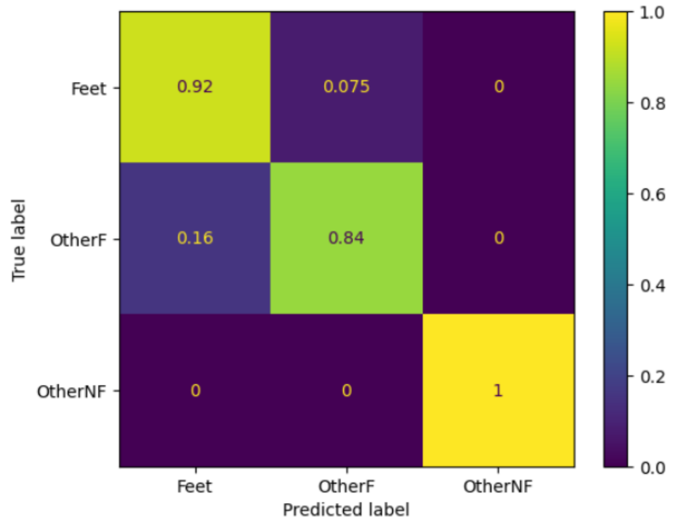


Fig. 22: Confusion matrix with three classes for a test data set.

X. Conclusion

This master's thesis paper aimed to create an algorithm that can recognize a touch with any permissible body parts in football (soccer). The algorithm would use the data gained from an object detection software *YOLOv8* and evaluate that data with the features engineered for the same goal. The algorithm is trained on a set of videos gathered before making this thesis. The structure of the algorithm proved to work best when set up as a deep neural network with LSTM layer at its core. The conclusion drawn from the process is that a better object detection model is required that is more robust as numerous issues stem from dropping the objects of interest, such as an active ball or the active player in the frame.

References

[1] https://dataconomy.com/wp-content/uploads/2022/11/SAOT-AI-based-new-semi-automated-offside-technology-started-to-use-in-the-2022-V jpg
[2] https://x.com/FOS/status/1802768711293469040/photo/1
[3] https://i.ytimg.com/vi/Cc_zaUbF4LM/maxresdefault.jpg
[4] https://docs.ultralytics.com/quickstart/#install-ultralytics
[5] https://www.statsperform.com/opta/#:∼:text=Opta's%20men's%20and%20women's%20football,%2C%20more%20accurately%2C%20and%20faster
[6] https://d2l.ai/chapter_recurrent-modern/lstm.html