Identify Plays Based Upon Video Footage

Karam Shbeb
Innopolis Uinversity
k.shbeb@innopolis.university

Mohammad Shahin
Innopolis Uinversity
m.shahin@innopolis.university

Mahmood Darwish
Innopolis Uinversity
m.darwish@innopolis.university

***Abstract*—This document presents the project for Practical Machine Learning and Deep Learning course. Our project aims to detect football (soccer) passes, including throw-ins and crosses.**

I. Description

Currently, football passes in games are tracked manually, a person tasked with monitoring the passes keeps track of the passes played in a game. This is an expensive and time consuming process. This project aims to address that issue. Our work will help scale the data collection process. Automatic event annotation could enable event data collection from currently unexplored competitions, like youth or semiprofessional leagues or even training sessions.

In this project, we shall create a system that is capable of detecting the movements of the ball and detect when a pass, throw-in, or a challenge happens during the game. A pass happens when a player gives the ball to another on their team, a throw-in happens when the ball leaves the pitch and one of the players uses their hands to throw it back in, a challenge happens when two players are competing for the ball and the ball is in neither players' possession. Since players in football are allowed to run or walk with the ball, then not every movement of the ball is classified as a pass. Depending on the speed of the ball and its distance to other objects (players or borders of the pitch) the system will decide whether this is a pass or not.

II. Related Works

Nowadays football analysis companies hire full-time and part-time employees to watch all matches and collect the exact data from them. It happens because AI tracking systems can not extract data as much and are as accurate as people do. However, Radha Penumala and others [1] made a system to address that issue but their system only detects goals and does not detect more specific events such as passes.

A more useful system is mentioned in [2]. Where Yutaro Honda et al. discribe a system that can record passes. However, their system suffers from multiple limitations. The system doesn't detect long passes well, the data used to train the system almost only consists of short distance passes so the data is biased and is not representative. The system also doesn't monitor goal keepers and their passes have to counted manually.

A more promising system is mentioned in [3] by Sanjay Chawla et al. However, this system was trained with a very small dataset (only 4 matches) and has a lot of bias in the training data as all of the matches are of the same team (Aersnal) in their home turf during the same season and Aersnal won in all of those matches. This means that the model probably did not get introduced to different styles and strategies and cannot be generalized to other matches.

III. Methodology

To help in solving this problem a huge data set is made available in Kaggle's competition DFL - Bundesliga Data Shootout. This set contains train and test data serving the same purpose. Train data includes comprising video recordings from eight games. Both halves are included for four of the games, while only one half is included for the other four games. These videos are appended with events annotations including video id, type of event, descriptive attributes for the event, and the time in seconds when the event occurred within the video. In addition to training and testing, data-sets Kaggle provides clips of ten additional matches without event annotations and they can be used to help the model generalize to environments not represented in the training data.

A. Data description

In the dataset, there are three folders: *train, test*, and *clips*. Each of which contains mp4 videos. *train* is a folder containing videos to be used as training data, comprising video recordings from eight games. Both halves are included for four of the games, while only one half is included for the other four games. *test* is a folder containing videos to be used as test data. The test data for the public leaderboard of the competition comprises video recordings from one full game and four half games, the other half of each game being in the training set. This folder is only used for testing and evaluation purposes as the annotations are not provided. *clips* folder includes short clips from ten games, provided without event annotations. The purpose of these clips is to help models generalize to environments not represented in the training data. Lastly, we have the file *train.csv*. The file is event annotations for videos in the *train* folder. Below is the description of each of the columns.

- video_id - Identifies which video the event occurred in.
- event - The type of event occurrence, one of challenge, play, or throw-in. Also present are labels start and end indicating the scoring intervals of the video. See the Evaluation page for information about scoring intervals. Scoring intervals are not provided for the test set videos.

- event_attributes - Additional descriptive attributes for the event.
- time - The time, in seconds, the event occurred within the video.

1) Event types and Event Attributes: There are mainly five types of events, and 11 possible event attributes. We briefly describe them in this part. If more details are needed, please refer to the competition Event Descriptions page

- Plays: A Play describes a player's attempt to switch ball control to another member of his team. A play event may be executed as a Pass or as a Cross. It has the following possible attributes: [Pass, Cross], [Open Play, Corner Kick, Free Kick]
- Throw-Ins. The possible attributes are: Pass and Cross.
- Challenge: A Challenge is a player action during which two players of opposing teams are physically capable of either gaining or receiving ball control and attempt to do so. The possible attributes for Challenge are the following: Opponent rounded, Ball action carried out, Fouled, Opponent dispossessed, Challenge during release of the ball, Possession retained during challenge.
- The event type also includes start and end indicating the scoring intervals of the video. Please refer to the Evaluation page in the competition for more details. 'train.csv' contain 11218 records.

B. Data exploration

In this part, we go further than the generic shape of the data to analyze the data at hand and potentially discover underlying properties and correlations. We will use statistical tools in order to analyze the data and find insights that might be useful for us in the subsequent sections and in the following data mining steps.

1) Exploring the video id: The training set contain 12 videos. The duration of all videos are almost an hour. The frame rate is 25 for all training set videos. Figure 1 shows the total number of events by video. We can see from the figure that the video have fluctuating number of events associated with them. The minimum is 737 while the maximum is 1249. This comes as a result of the different styles teams have. For example, a team may focus more on possession which increases the number of passes.

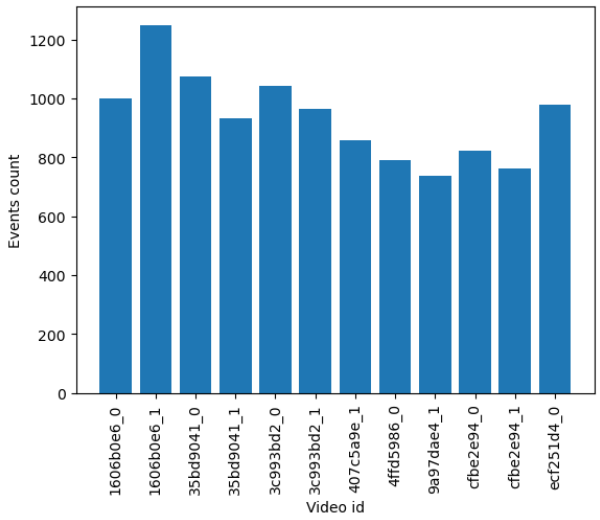

Fig. 1. Videos to total number of events

2) Exploring the event type: As we can see in figure 2, the majority of events is distributed between the events: end, play, and start. The challenges represent only 5% of the events. The throw-ins are even more rare as they cover only 1.5% of the total dataset.

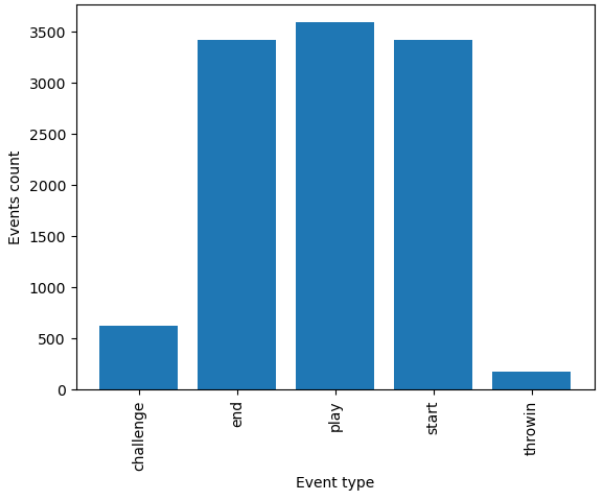

Fig. 2. Events count for each event type

3) Exploring the time: To properly visualize the time-series data, we divided the time into segments, each of 20s duration. This enabled us to plot figure 3 and infer that events reach its peak at the middle of the match half. Please notice that the small values (close to zero) at the beginning and the ending is due to the fact that the half has ended, or has not even started. There are no indicators in the dataset when halves start or end.

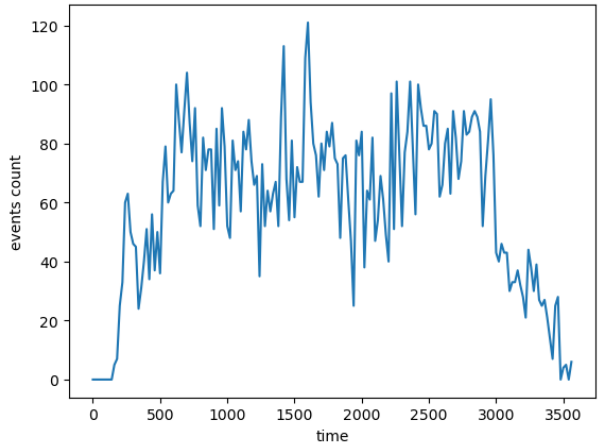

Fig. 3. Events count at each second

4) Exploring the intersection of events: To understand the cases where more than one event happens at the same time we counted the instances of intersection of events in the training data. More precisely, we looked at the cases where there's more than one event happening between a start and end events. This will allow to understand the likelihood of intersections happening and account for that in our model. Figure 4 shows ours findings

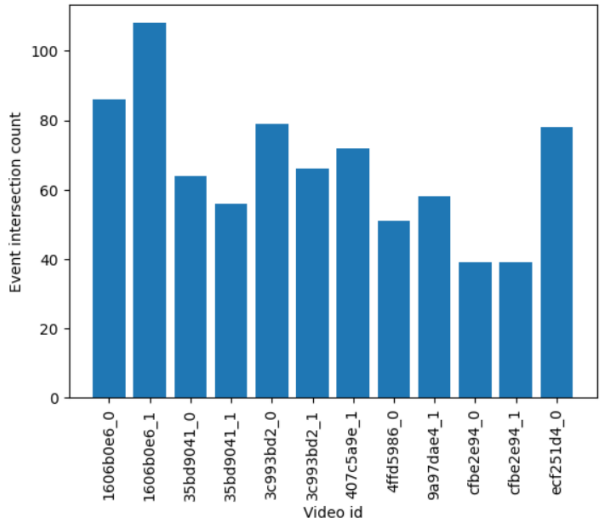

Fig. 4. Intersection of events in every video

5) Exploring intersection types: Understanding intersection doesn't only mean knowing how much they occur but also which types of events are more likely to occur at the time. As it shows in the figure 5 the majority of cases where an intersection happens it is between a play and a play type of event and we also notice that no two throw events happen at the same time.

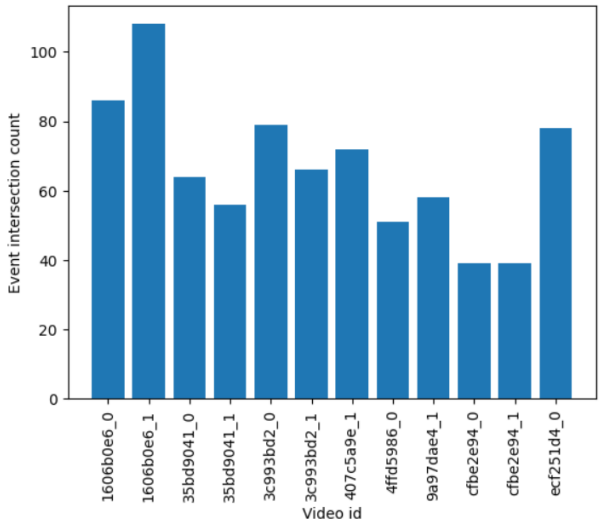

Fig. 5. Counts of types intersection

6) Exploring the event attributes: The attributes weren't deeply analyzed because the model is not dependent on them and is not expected to predict them.

C. Data pre-processing

We started pre-processing our data to make it easier to work with by first chopping the videos into the frames that make them up. The following graph 6 shows how many frames are in each video.

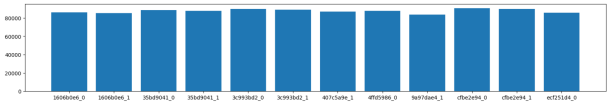

Fig. 6. Counts of frames

1) Creating the training set: During our work we first needed to create the training set. First, instead of looking at events as if they happen in an exact timestamp, we created a small range around the event timestamp. We took all the frames in that range and added them to a folder with the same name. At the end, we would have 4 folders each folder will be filled with the images from frames where an event was happening. It is true that we have 3 events to catch (passes, throw-ins, and challenges) but we also added a 4-th kind of event to be caught. That event is "background". Basically, every frame that doesn't belong to an event will be added to the event "background". Because we need to train our model on the frames where no event is happening too. That way we transform the problem into a 4 class classification problem. We will input for the model a time series of photos (a video) and for each photo it will predict which one of the 4 classes it is.

IV. GitHub link

The link to the GitHub repo can be found here.

V. Experiments and Evaluation

In this section we will talk about the experiments that we have preformed and how our model evolved during the process. We used mean average precision as our evaluation metric as it is common in object detection problems. We also used Kaggle's code to calculate mAP since it is open source.

A. Experiment I: Base-line Model

At first we wanted to understand how an already made model will preform when being fed the raw data without much in terms of modifying the model or editing the data. We wanted to create a base line model to compare our work to. Therefore, we decided to use EfficientNet model [4], specifically the "tf efficientnet b5 ap" implementation from the "Timm" library.

1) What is EfficientNet model: EfficientNet is a convolutional neural network architecture and scaling method that uniformly scales all dimensions of depth/width/resolution using a compound coefficient. Unlike conventional practice that arbitrary scales these factors, the EfficientNet scaling method uniformly scales network width, depth, and resolution with a set of fixed scaling coefficients.

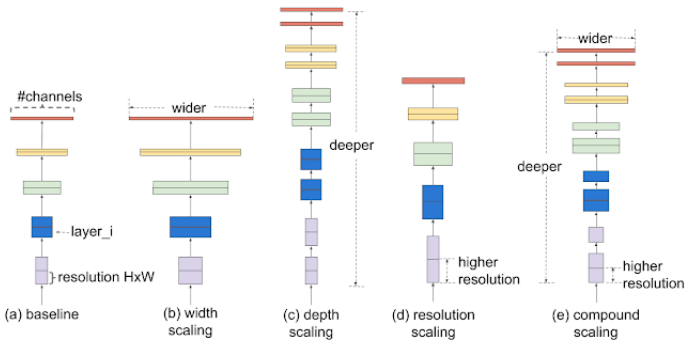

Fig. 7. ImageNet structure

2) Why EfficientNet: We use EfficientNet as our baseline model because they provide better capacity for more data [6] and we have relatively large data in our training dataset. Despite being suitable for large data EfficientNet surpasses many competitors in terms of accuracy such as VGG, ResNet, AlexNet, Ierception v4, SqueezeNet, and DensenNet [7]. According to the author of EfficientNet [4] the architecture of this model can be sorted into 7 groups (B0 to B7). Even though B6 achieves the second highest accuracy after B1 [4] we chose B6 because it's a light version of B7 and it's robust against adversarial attacks [5]. Choosing a robust algorithm is crucial in our case because our model is designed to run outdoors with inconstant intensity and lighting.

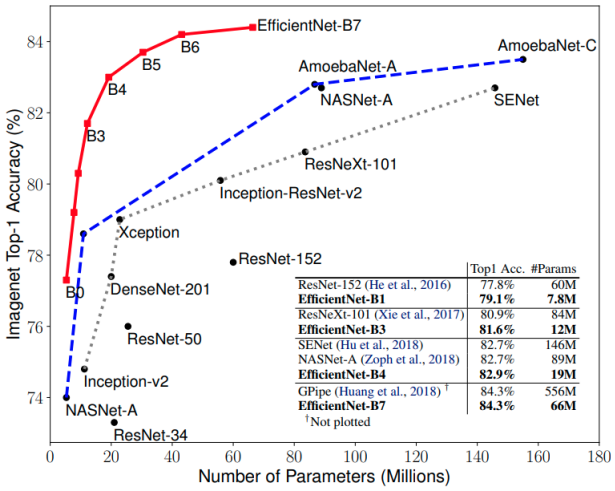

Fig. 8. Accuracy of ImageNet models

3) Training the base-line model: Since we split the video into frames and pass it into the model we turned the problem into 4 classes of classification (background, challenge, play, and throw-in). We split our training data into both validation and training data and feed it into the model. Since we are using a pre-trained model from "Timm", we train our model using this library which provides checkpoints. Therefore, we don't need to train the whole model continuously, and we can train each epoch or more separately. Timm provides a script that takes the checkpoints and averages the weights from the last few ones using the script provided by Timm to save it as a model. This feature is very important in our case because we don't have suitable hardware to run the model continuously.

4) Testing the base-line model: Our testing data consists of video recordings from one full game and four half-games, the other half of each game being in the training set. In testing phase we use the same scheme that we used in training phase, we split our video into frames and we match it with the events using the timestamp written on the labeled test data. We run our model on the testing data and we extract the predictions of our model in order to compare it with the real labels. Firstly, we extract both scoring intervals and ground-truth events. Secondly, we map each event class to its prevalence and we create a table of detections with a column indicating a match to a ground-truth event. Thirdly, we remove detections which occur outside of scoring intervals. Finally, we compare our predictions with the real labels taking tolerance into account. The accuracy of our model is 7% which is very low but expected from a model trying to solve this complex problem without any data normalization or change in format. We will compare the results of the next experiments with the base-line model to see the improvements our changes make.

B. Experiment II: Absdiff

To improve the results from the previous model we changed the training input. We replaced the i-th frame with the absolute difference between the i-th and i-1-th frame. Then the input

for the i-th frame because the concatenation of 11 frames. We inputted 5 frames before the i-th frame + i-th frame + 5 frames after the i-th frame. These 11 frames (which are modified by the absolute difference) were the input for the i-th frame. The intuition behind this idea is that between two consecutive frames if there's an event then both frame will be equal except for the place of the ball, because during an event the ball will move way faster than the players or other objects in the photo. So, subtracting one frame from the other will make the whole photo black except for the ball if there's an event. Adding 11 frames will help the model figure out which type of event it is because in those 11 frames the movement of the ball will be extremely highlighted compared to other objects in the photos and the model can use that movement to figure out the type of the event. Testing this model gave an accuracy of 43%.

1) Resolution enhancement: To improve the previous model we used a technique common in computer vision and resized the photos to a higher resolution. From 456×456 to 1024×576 resolution. That improvement increased the accuracy to 49%.

C. Experiment III: Gray scaled concatenation

Since the results of the previous experiment were promising we decided to use the same method of using multiple frames and doing some image modification to highlight the position of the ball. We first turned the videos into frames, then gray scaled every frame. To train the model we turned the input of the i-th frame to be the concatenation of (i-1)-th, i-th, and (i+1)-th frames channel wise. This caused the input to not have any artifacts when the nothing is moving too fast on the pitch. However, in the cases where the ball moves very fast (usually it means that an event is happening) we get frames that look like the following figure.

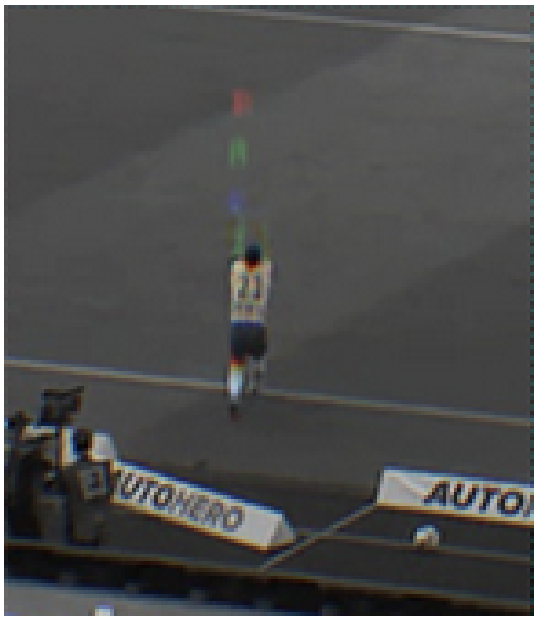

Fig. 9. Input image after gray scale concatenation

Using this method and up-scaling the image to a high resolution (similar to experiment II) we got an accuracy of 61%.

1) Change to tf efficientnet b0 ap: After experimentation it turned out that "tf efficientnet b0 ap" model preforms better than "tf efficientnet b5 ap". Changing to "tf efficientnet b0 ap" as our back bone improved the performance to 68%.

VI. Analysis and Observations

It is really interesting that the EfficientNet did not score well on its own. It seems that not having any temporal information makes it impossible to detect and classify events. In fact, even for humans it might be very difficult to detect if a pass is going on from a still image of a football game. Therefore, it seems that the idea of concatenating multiple frames to provide some information about what is changing as time goes back is crucial for the performance of the model. Another thing that might have affected the result greatly was highlighting where the ball was for the model. It seems that those were the most important ideas for the model to give better results.

VII. Timeline

In the first two weeks of September we chose the idea, collected data, split the project into stages and planned what we are going to do. In the next two weeks we started analysing the data and doing research to find the best model that can solve our task. In October, we started doing experiments after choosing EfficientNet b6 as our model. We created a training dataset, trained a base-line model and improved the model by applying *absdiff* to the training data. During the first two weeks of November, we proposed another idea for frames extraction and processing which is gray scale concatenation. The previous idea made a great impact on the accuracy of our model. During the last two weeks of November, we wrote the technical report and replaced EfficientNet b6 with b0 which had a better score.

VIII. Contributions of participants

The idea was proposed by Karam, he also did the gray scale concatenation. Mohammad did data pre-processing and research on model selection. Mahmood did *absdiff* experiment and baseline model. Writing the technical report and other experiments and coding steps were done by all individuals in the team.

IX. Conclusion

We described a new model to detect passes, throw-ins, and challenges automatically in football games. This system relies on temporal information and the highlight of the position of the ball. We presented the experiments that we have done to reach this model along with the results and modifications of those experiments to get better results.

REFERENCES

[1] R. Penumala, M. Sivagami, and S. Srinivasan, "Automated Goal Score Detection in Football Match Using Key Moments," Procedia Computer Science, vol. 165, pp. 492–501, January 2019

[2] Y. Honda, R. Kawakami, R. Yoshihashi, K. Kato and T. Naemura, "Pass Receiver Prediction in Soccer using Video and Players' Trajectories," 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2022, pp. 3502-3511, doi: 10.1109/CVPRW56347.2022.00394.

[3] Sanjay Chawla, Joël Estephan, Joachim Gudmundsson, and Michael Horton. 2017. Classification of Passes in Football Matches Using Spatiotemporal Data. ACM Trans. Spatial Algorithms Syst. 3, 2, Article 6 (June 2017), 30 pages. https://doi.org/10.1145/3105576

[4] Mingxing Tan and Quoc V. Le. "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks". In: (2019). DOI: 10.48550/ARXIV.1905.11946. URL: https: //arxiv.org/abs/1905.11946

[5] Cihang Xie et al. "Adversarial examples improve image recognition". In: Proceedings of the IEEE/CVF Confer- ence on Computer Vision and Pattern Recognition. 2020, pp. 819–828.

[6] Qizhe Xie et al. "Self-training with noisy student im- proves imagenet classification". In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020, pp. 10687–10698.

[7] Pan Zhang, Ling Yang, and Daoliang Li. "EfficientNet- B4-Ranger: A novel method for greenhouse cucumber disease recognition under natural complex environment". In: Computers and Electronics in Agriculture 176 (2020), p. 105652.