Soccer Events Detection

Federico Piscitelli[970949]

Dipartimento d'Informatica, Università degli Studi di Milano.

Contributing authors: federico.piscitelli@studenti.unimi.it;

Abstract

In the world of sports analytics, the ability to automatically detect and identify events within a soccer match holds immense potential for enhancing coaching strategies and player performance evaluation. This project aims to develop a system to identify soccer and non soccer events and classify relevant soccer events from images, extract from match footages. Leveraging convolutional neural networks (CNNs) and image processing algorithms, the system analyzes video frames to recognize various events such as penalties, tackles, corner kicks, cards and more. The study also compares two different CNN and their relative performances, evaluating the capabilities and the struggles of each one

Keywords: Football event detection, deep learning, image analysis

1 Introduction

The aim of the project is to built an automatic system that is capable of distinguish between non soccer and soccer photos with the respective category of event (penalty, free kick, corner, tackle, substitution, cards, yellow cards, red cards) or part of the pitch (right, center, left).
To achive the final goal a CNN was constructed, leveraging deep learning techniques to classify images and discern complex details within soccer-related scenes.

1.1 Dataset

In order to be able to train the model over different image samples, a new dataset was created. Initially, the original dataset was partitioned into distinct segments for training and testing purposes, and for each segment it was divided into different categories of events or part of the pitch. To augment the dataset with non soccer images, a small

portion of an additional dataset containing general images was used. To ensure accessibility and integration with the Google Colab environment, the datasets were uploaded to Kaggle, enabling convenient retrieval for subsequent experimentation and analysis. An important consideration when constructing and training a CNN for multiclass image classification is the distribution of the dataset. This is crucial because an imbalanced dataset could lead to potential issues of overfitting or underfitting.
A portion of the training dataset is also set aside for validation during training. This allows to monitor the model's performance and prevent overfitting.
Below [Fig. 1] are two images that show the distribution of the datasets (training and testing) across the various classes:

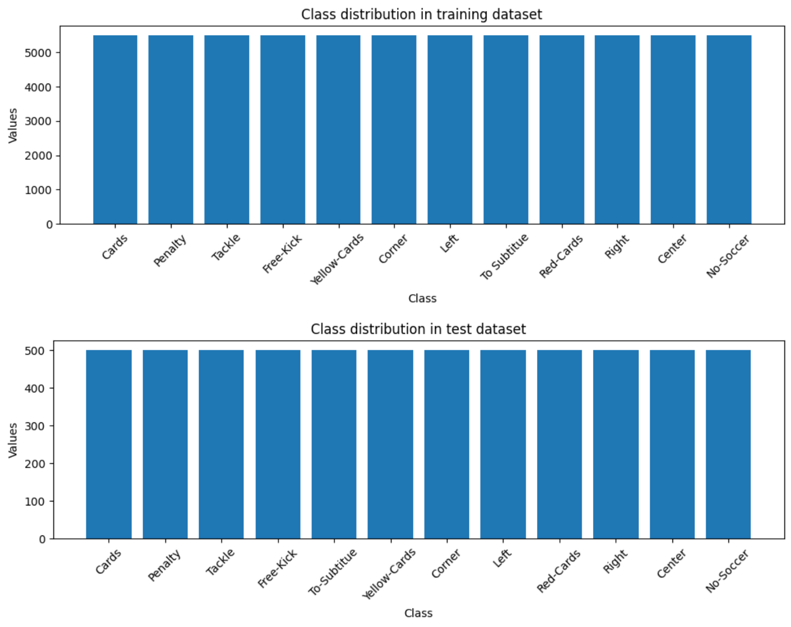


Fig. 1: Distribution of classes in training and test dataset

Before using any of the images in the dataset, their quality was checked: if a certain quality threshold was not met, the image was removed from the folders. Below [Fig. 2] are some sample images:

2 Experimental results

In the following chapter, we present the results obtained from the application of two distinct convolutional neural network (CNN) models. These models were trained and evaluated on the dataset previously mention. The chapter provides insights into the performance of each model, including metrics such as accuracy, precision, recall, and F1-score. Additionally, we analyze the strengths and weaknesses of each model and

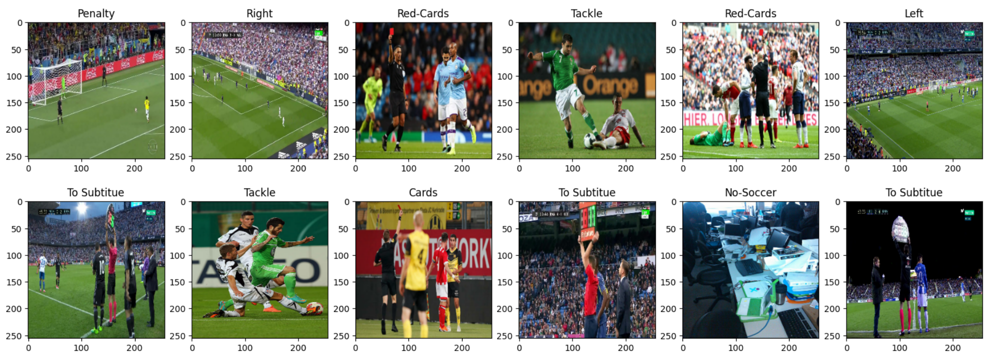


Fig. 2: Sample images for different classes

discuss how their architectures and parameters influenced their respective outcomes. Through a comparative analysis, we aim to highlight the effectiveness and limitations of the different CNN approaches in accurately classifying soccer-related images.

2.1 Structures of CNNs

The two CNNs constructed are quite similar in terms of structure: they differ in one layer and in the number of filters applied by the convolutional layers. It could be assumed that a network with more layers might be more accurate, but in reality, a greater number of layers, especially considering the type of images being analyzed, could lead to overfitting.
The first network [Fig. 3] is built with 3 convolutional layers, to which respectively 32, 64, and again 32 filters of size 3x3 are applied. After each convolutional layer, a pooling layer has been inserted, which reduces the size of the extracted features, in the case of the examined CNNs, by creating a 2x2 matrix containing the maximum values of the features extracted from the previous layer.
The second network [Fig. 4] is constructed with two convolutional layers, to which 128 and 64 filters are applied respectively. Again, in this case, a pooling layer has been inserted after each convolutional layer.
Both networks start with a preprocessing layer for images to increase the diversification of the image types in the dataset and prevent overfitting. This preprocessing layer includes:

- rescaling: rescales the pixel values of the input images to a range between 0 and 1.
- random zoom: randomly applies zoom augmentation to the image
- random translation: randomly translates (shifts) the images horizontally and vertically.
- random rotation: apply a random rotation to the image

Both networks end with:

- a flattening layer, which flattens the output from the previous layer into a 1D array, preparing it for input into a fully connected dense layer

- a dense (fully connected) layer with 256 neurons
- a dropout layer that randomly sets the output of 50% of the neurons to zero, to prevent overfitting
- a final dense (fully connected) layer with 12 neurons (equal to the number of output classes), using the softmax activation function to output class probabilities

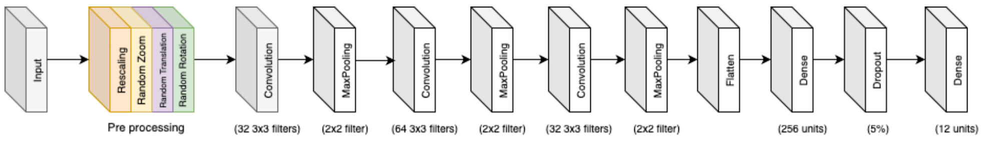


Fig. 3: Structure of the first CNN

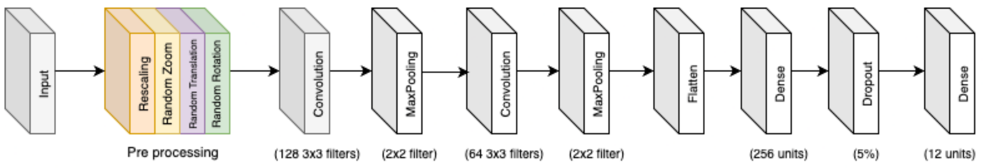


Fig. 4: Structure of the second CNN

2.2 Training

Both CNNs were trained over 20 epochs, during which they were provided with the valid images from the dataset. Throughout this phase, the training progress was monitored using metrics such as accuracy and loss. Upon completion of the training phase, precision, recall, and F1-score were computed, and a confusion matrix was generated. These metrics offer valuable insights into the models' performance in accurately classifying images across the different classes while minimizing errors. Additionally, to enhance training efficiency and model performance, several TensorFlow callbacks were implemented:

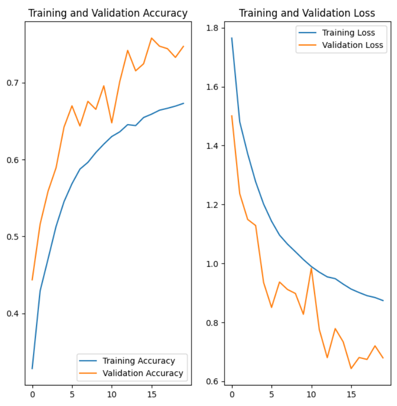
- Reduce Learning Rate: adjusts the learning rate if a certain metric fails to show improvement over a specified number of epochs.
- Early Stopping: halts the training process if the accuracy fails to increase for a specified number of consecutive epochs, thereby preventing overfitting.
- ModelCheckpoint: saves the model's weights whenever an improvement is observed compared to the performance in a previous epoch. It is important to note that this callback was primarily used to create backups of the model, rather than to enhance performance.

Through experimentation and tuning, the CNN models were trained to achieve a good level of performance, balancing accuracy and generalization across the dataset's diverse classes.

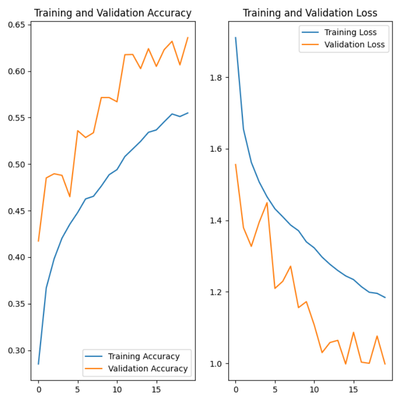
2.3 Metrics

2.3.1 Accuracy and Loss

The graphs depicted in Figure 5 show the accuracy and loss during the training phase (blue line) and validation phase (orange line). It can be observed that the first model generally have high accuracy and lower loss, despite using fewer filters.
Both models appear to exhibit some degree of overfitting, as indicated by the increase in the loss value during the validation phase. This suggests that the models perform well on the training data but struggle with unseen data. This phenomenon may be attributed to the limited amount of data or its lack of variation.



(a) 3 layers CNN



(b) 2 layers CNN

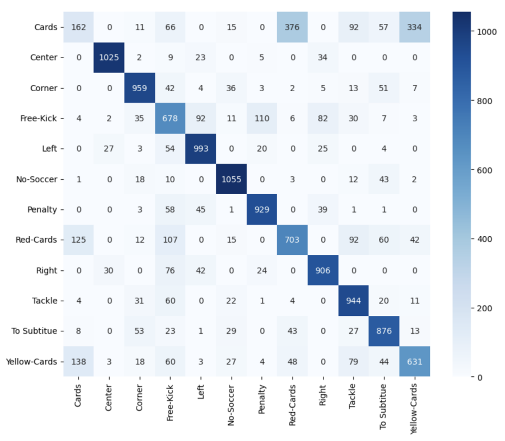
Fig. 5: Accuracy and Loss for the two CNN

2.3.2 Confusion Matrix

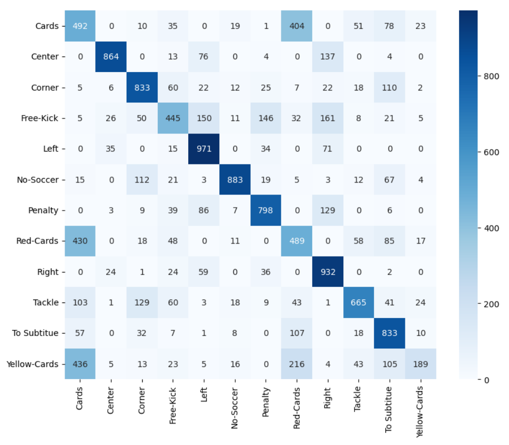
In Figure 6, the two confusion matrices of the 2 models are depicted: the correct values are represented on the y-axis, while the predicted values are shown on the x-axis. Certainly, some observations can be made regarding the performance of the two models:

- The CNN with 3 layers demonstrates a higher level of accuracy in predicting new images.
- Both models struggle somewhat to differentiate between various types of cards, with the 2-layer CNN exhibiting more generalization in card classification.

- The models tend to confuse a ”card” event with a ”substitution” event, likely due to the similar positions of the figures in the photos: two actors involved, with one of them holding something colored above their head.
- For a similiar reason as above the models tend to confuse free kick with penalty and viceversa



(a) 3 layers CNN



(b) 2 layers CNN

Fig. 6: Confusion matrix for the two CNN

2.3.3 Precision, Recall, F1-Score

As reported in Table 1 Model 1 consistently outperforms Model 2 across all metrics, indicating that the additional layer in Model 1 contributes significantly to its better performance in detecting football events in photos. With higher precision, recall, accuracy, and F1 score, Model 1 demonstrates its superior capability in identifying football events across various classes compared to Model 2. However, despite its simplicity, Model 2 struggles with recall, implying that it misses a considerable number of actual positive events in the dataset

Model	**Precision**	**Recall**	**Accuracy**	**F1-Score**
Model 1	0.84	0.65	0.74	0.73
Model 2	0.81	0.47	0.63	0.62

Table 1: Classification report for Model 1 and 2

The Table 2 presents the same metrics analyzed previously, but divided by class. From this table as well, we can observe that overall Model 1 performs better than Model 2 in almost all classes.

Class	Precision		Recall		F1-Score		Support
	Model 1	Model 2	Model 1	Model 2	Model 1	Model 2	
Cards	0.366516	0.318859	0.145553	0.442049	0.208360	0.370482	1113
Center	0.942962	0.896266	0.933515	0.786885	0.938215	0.838021	1098
Corner	0.837555	0.690141	0.854724	0.742424	0.846052	0.715328	1122
Free-Kick	0.545455	0.563291	0.639623	0.419811	0.588797	0.481081	1060
Left	0.825436	0.705669	0.881883	0.862345	0.852726	0.776179	1126
No-Soccer	0.871181	0.896447	0.922203	0.771853	0.895966	0.829497	1144
Penalty	0.847628	0.744403	0.862581	0.740947	0.855039	0.742671	1077
Red-Cards	0.593249	0.375288	0.608131	0.423010	0.600598	0.397723	1156
Right	0.830431	0.638356	0.840445	0.864564	0.835408	0.734437	1078
Tackle	0.731783	0.761741	0.860529	0.606199	0.790951	0.675127	1097
To Subtitue	0.753224	0.616124	0.816403	0.776328	0.783542	0.687010	1073
Yellow-Cards	0.604986	0.689781	0.598104	0.179147	0.601525	0.284424	1055

Table 2: Classification report for each class for Model 1 and 2

2.3.4 The CNN at work

Figure 7 contains several examples of images with their respective predictions by Model 1, demonstrating its ability to identify the football event (or non-event) associated with the analyzed image.

3 Concluding remarks

In conclusion, the objective of the project, to build a CNN capable of identifying football events, has been achieved. We observed that a low number of layers can prevent overfitting, but it can also make the model much less accurate, as seen in Model 2.
Further optimizations and fine-tuning could be explored to enhance the performance of both models. Additionally, expanding the dataset with more diverse and representative images could help improve the models' ability to accurately classify football events. Furthermore, investigating alternative architectures and techniques could lead to the development of even more efficient and accurate models for information retrieval tasks in the context of sports events.

References

[1] Ali Karimi, Ramin Toosi, Mohammad Ali Akhaee (2021) *Soccer Event Detection Using Deep Learning*

[2] *Unsplash random images collection* on kaggle.com

[3] Tensorflow

[4] Google Colab

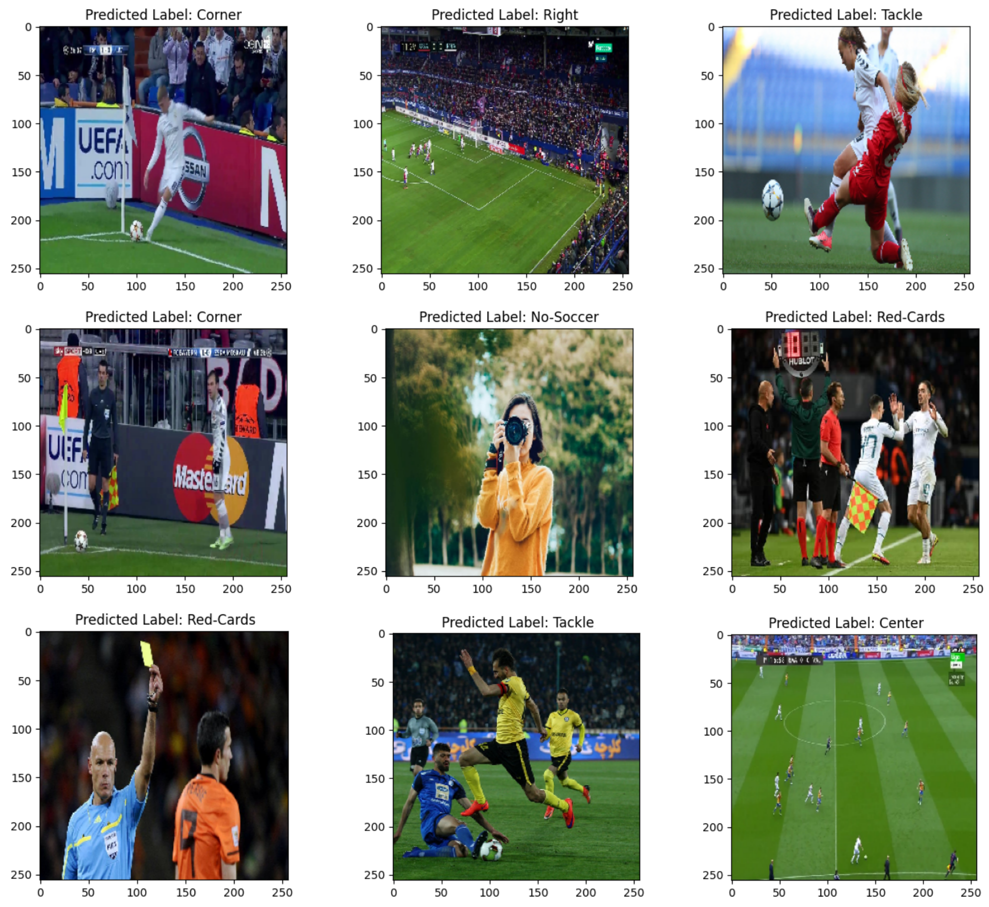


Fig. 7: Sample predictions made by Model 1