Automatic detection of events in sports videos

Vinayak Parab

Supervisors:
Dr. Prof. Simon Ziegler
Dr. Prof. Franz Hollich

Thesis submitted for the degree of Master in
Big Data and business Analytics
18 credits

Department of Informatics
Faculty of Information, Media and Design

15th September 2021

Automatic detection of events in sports videos

Vinayak Parab

Automatic detection of events in sports videos

https://moodle.hochschule-heidelberg.de/my/

Affidavit

I, Vinayak Sitaram Parab, Herewith declare:

- that I have composed the chapters for the Master Thesis for which I am named as the author independently,
- that I did not use any other sources or additives than the ones specified,
- that I did not submit this work at any other examination procedure.

Signed: _____

Date: _____

Acknowledgment

I would like to thank my supervisors, Simon Ziegler, Franz Hollich, Pål Halvorsen and Michael Riegler for their guidance and contributions, and Vajira Thambawita for all the help along the way. I would also like to thank my family and my friend Shubham Shinde for all the encouragement and patience throughout the process.

Abstract

Numerous different approaches have been used to perform automatic event detection in the past few years. For soccer games, a summarized clip comprises of game highlights which covers the key moments in a soccer game. At present, to create such game highlights, the sports events are manually annotated by human operators. It is not feasible to annotate these events manually as it takes large amount of time and human intervention. Recent research has shown that deep learning techniques can be used to find these events without any human intervention. In order to avoid computational complexity of training the deep learning networks with raw data, most approaches use pre-computed features extracted from raw images for training purpose.

This thesis presents a fine-tuned approach of a deep learning approach that can automate the manual annotation of events in sports videos. We experiment with pre-computed features extracted from single deep learning model and pre-computed concatenated features extracted from multiple action recognition models for the SoccerNet-v2 [17] dataset. We compare the results to state-of-the-art models for the task of action spotting on the SoccerNet-v2 [17] dataset. We also analyze the impact of fine-tuning the selected hyparparameters in the respective the deep learning approaches.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Nowadays, it is easily possible to get access to plethora of high-quality video content through subscription-based streaming services like Netflix, Amazon, etc. In today's fast paced world, it is not easy to keep track of detailed information about everything that is happening around the world. Due to this, creating a compact summary of key events from a long untrimmed clip has gained lot of popularity. Soccer is the most watched sport in the world and is enjoyed by its fans across the world [56]. In the FIFA world cup 2018, FIFA decided to use a video assistant referee system (VAR) [67] which uses computer vision technology. VAR[67] assists the referee to determine 4 different type of incidents that have been identified as game-changing decisions. VAR [67] highlights the use of video summarization techniques. Today, there are several companies who provide on-demand subscription-based live online streaming services and video highlights which includes a short summarized clip of key moments from the game. The content is consumed by the subscribers on their computers and smartphone devices. In 2019, it was reported that 47.1% [4] of the private households were estimated to have a computer at their home. In the year 2020, the total number of smartphone users worldwide crossed the figure of 3 billion [52]. With increasing numbers of gadgets day by day, it is extremely crucial for these companies that they ensure the timely distribution of sports events/highlights. For

soccer or sports events in general, a summarized clip would comprise of highlights such as goals, bookings, goal attempts and penalties. At present, to create such game highlights, the sports events are manually annotated by human operators. It is a tedious process to annotate these events manually. Automating the process of annotation would have a large impact on the games played in the lower division of the football hierarchy where the funds are limited.

Multiple different type of deep learning algorithms have been used to solve the problem of automatic event detection. This thesis aims at developing a system that can automate the manual annotation of events like tackles and other controversial events in sports videos and at the same time is cost effective and scalable. Considering the recent advancements in the field of Artificial Intelligence, a potential solution could be to make use of a deep learning model given that it provides pre-built packages/frameworks for analyzing digital images and videos. This data could be further used for statistical analysis purpose, which in turn could provide more value to fans, teams and broadcasters of the sport events.

1.2 Problem Statement

Our goal is to use a deep learning model and video processing techniques which will detect and classify events like tackles, mistakes, etc. in soccer videos. The system should be able to detect and classify the events correctly. The approach would be to make use of a deep learning model which will generalize well for other sports events and can be scaled easily if needed. The selected approaches were evaluated for the task of action spotting. Action spotting provides a particular frame in a video along with the timestamp at which an event has occurred. This thesis aims to answer the question:

How to perform automatic detection of video events in sports videos?

Below objectives were defined based on the research question:

1. Research and develop novel approaches for action spotting and action classific-

ation in soccer videos.

2. Analyze the performance of the approaches designed in step 1 using a benchmark dataset like SoccerNet-v2 [17].

3. Perform a comparative study of the deep learning approaches proposed in the above steps for the task of action spotting and compare their results to state-of-the art architectures.

1.3 Scope and Limitations

There are multiple challenges involved with the task of automatic event detection and recognition. One of the challenge is limited availability of datasets. During this thesis, the recently published SoccerNet-v2 [17] will be used which is the extended version of the SoccerNet dataset[32]. The extended version SoccerNet-v2 contains a total of 17 different classes that could be used for the task of action spotting. The scope is limited to the task of action spotting wherein the final goal is to make use of a deep learning model to predict a frame in a video along with its timestamp at which an event has occurred. There are multiple traditional approaches involving traditional computer vision techniques. However, the scope is limited to using deep learning models.

1.4 Research Method

A research problem could be tackled in multiple ways. The research method provided in the Association for Computing Machinery's (ACM) research methodology was followed. In 1989 ACM's Education Board released the report 'Computing as a Discipline' [19]. Below are the three paradigms on which our work is based on :

- *Theory* The theory paradigm is based on mathematics and comprises of four steps adhered in the developing a coherent and a valid theory. These steps are (i) Characterization of objects of study (definition), (ii) Hypothesization about finding possible relationships among them (theorem), (iii) Discover whether these relationships are true, and (iv) Interpretation of results. [19]

- *Abstraction* The abstraction paradigm consists of experimental scientific method and comprises of four steps. These steps are (i) Establishment an hypothesis, (ii) Construction of a model for prediction purpose, (iii) Conceptualization of an experiment and collecting the relevant data for it, and (iv) Inspection of the results.

- *Design* The design paradigm design is based on engineering and comprises of four steps followed to solve a given problem. These steps are (i) State the requirements, (ii) State the specifications , (iii) Design and implement the desired system (iv) Test the developed system.

Various approaches were researched and applied for soccer detection and analyzing the results. The fine-tuned approaches were designed and validated against SoccerNet-v2 dataset[17]. This supports the design paradigm well.

1.5 Main Contributions

The performance of deep learning approaches was assessed for understanding the semantic context in a video input and the network was fine-tuned with the selected hyperparameters to see if that improves the performance. The impact of training these approaches was tested on features extracted from ResNet-152 and pre-computed concatenated features obtained by concatenating the features from 5 different models as illustrated in 3.3. Following contributions were made in the context of the thesis:

- Different machine learning approaches were researched for event detection and various experiments were performed with the deep learning models selected for these task of action spotting. These experiments were performed with SoccerNet-v2[17] dataset which includes 17 different classes with 300k temporal annotations over soccer broadcast videos. The existing models were fine-tuned by optimizing the selected hyperparameters in 3.4. The selected deep learning

models were tested with pre-computed features concatenated from 5 different 3D convolution models along the feature dimension.

- The experiments were performed with 2 different deep learning models on SoccerNet-v2 dataset [17].These approaches were tested for the task of action spotting and the performance was assessed for several metrics like accuracy,precision and average mAP. Different ablation studies were performed with various hyperparameters some of the parameters were fine-tuned in the deep respective models.

- The performance of the CALF method and the temporally aware pooling method was compared on both set of features. It was observed that there was no significant increase in the performance of the CALF method when trained using both set of features. However, there was a significant increase of 2.5% for the average mAP metric in the performance of the Temporally Aware Pooling method when trained on concatenated pre-computed features as discussed in 3.4.

The experiments performed adhere to the information provided in the problem statement 1.2. Additional experiments were performed by fine-tuning some of the hyperparameters which help the selected deep learning approaches model the context better.

1.6 Outline

Chapter 2 - Background In Chapter 2, all the relevant concepts and terminology used in the field of machine learning and deep learning was described. Some of the preeminent related work and state-of-the-art deep learning approaches relevant for the task of event detection and action spotting were discussed.

Chapter 3 - Methodology In Chapter 3, the dataset was presented that will be used during this thesis. Further, the pre-processing and actionness detection techniques were discussed to filter out predictions below certain threshold. Multiple different

contextually aware deep learning approaches were discussed. Further, a detailed explanation about the selected deep learning techniques was provided. Further, the effect of using pre-computed concatenated features on the contextually-aware approaches was discussed.

Chapter 4 - Experiments and Results In Chapter 4, the results of all the contextually aware deep learning approaches were presented. Further, the effect of using pre-computed concatenated features on these models was discussed. The results were compared using different accuracy metrics and the performance of the different models was evaluated with different hyperparameters settings.

Chapter 5 - Conclusion In chapter 5, a summary of this thesis and its contributions is provided. Further, the possible future work with respect to the thesis was discussed.

Chapter 2

Background

2.1 Introduction

The end goal is to present a scalable deep learning model which could be used to perform the task of action spotting in soccer broadcast videos. The selected deep learning models are fed with pre-computed features extracted from various action recognition models along with the labeled data and then model would detect and predict the event and would also provide a point in time at which this event has occurred.

Our goal is to present a deep learning model that will automatically detect and classify abnormal events in a video and also predict a point in time. In this chapter, all the relevant terminology and concepts related to machine learning is explained. The related works section provides brief information about all the the state-of-the-art deep learning models and the datasets that have been used over the last few years.

2.2 Event definition

An event can be defined in numerous ways. An event can be defined as an activity that occurs at a point in time. It is quite a natural task for humans to detect and identify an event. However, different people could predict diverse start and end points for the same event. In our case, it should not be a major roadblock as the events are already tagged with the ground truth.

2.3 Video annotations

Manual annotation of events in sports videos is an expensive and time consuming process. The process requires the annotator to watch the entire soccer game and then tag the events manually. Forzify tagging [29] lets the users create events by tagging the events directly on the video stream. The tagging operation consist of two stages ore levels. At the first level, the operator can monitor one or four simultaneous games, where he has buttons for each team and an event it generated pressing the event type for a team and can press the publish button. Then, a second level operator can fine-tune the events with additional metadata, exact timestamps and quality assurance. This two-step tagging process ensures the smooth tagging and distribution of video events. The two-step tagging process is expensive, but if some part of this tagging process could be automated the amount of human intervention could be reduced which in turn could reduce the efforts required for manual tagging.

2.4 Action Recognition

Action recognition can be defined as the method of classifying an action in a video. Action recognition is sometimes also referred to as Activity Recognition. In this case, the input would be a trimmed video from the original untrimmed video. The end goal is to determine the list of occurrences of different actions from the input. Classifying actions come with numerous challenges the primary being the high computational and storage costs. The task of action recognition is illustrated in Figure 2.1

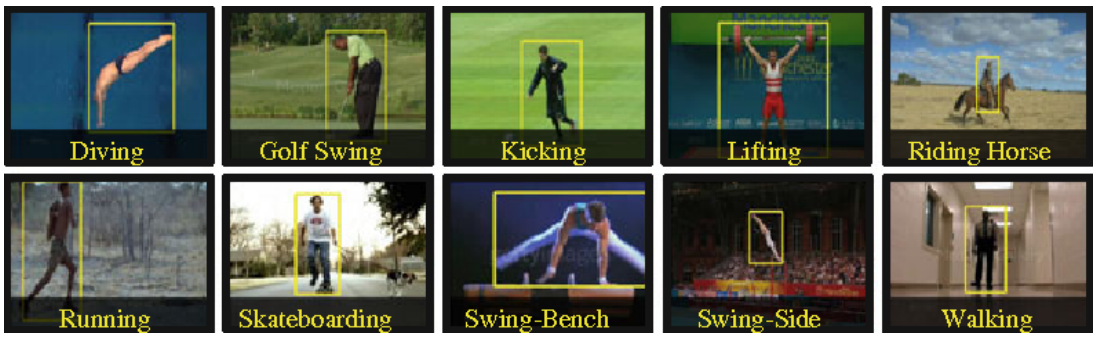

Figure 2.1: Action Recognition. Reprinted from [57]

2.5 Action Spotting

Action detection is the task of answering the question when a particular event has occurred. Action spotting consists of finding the timestamp of an event in a video clip [5]. This implies that there is a need to not only classify th event but also to provide the timestamp at which it occurs. Below Figure 2.2 illustrates the temporal localization of events.

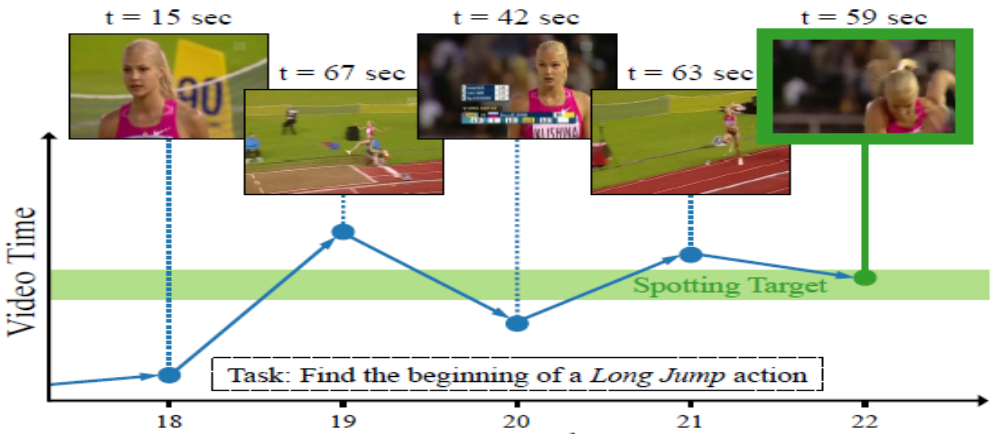

Figure 2.2: The figure shows the start and end time for the long jump. Reprinted from [5]

2.6 Machine Learning

Machine learning (ML) is a discipline of Computer Science which focuses on the use of data learns a mapping between the input and the output. Machine learning is the task of making a machine learn from input data and provide an output. The machine does the complex task of finding the mapping between input and output without being programmed.

2.6.1 Supervised Learning

Classification and regression algorithms are categorized as supervised learning techniques. Supervised learning techniques are capable of learning a mappable

function between the input and the output [2]. In case of classification and regression algorithms, the machine tries to find the mapping between the provided input data and the outputted data.

2.6.2 Unsupervised Learning

Clustering algorithms fall under the category of unsupervised learning techniques[2]. Unsupervised learning techniques learn are capable of learning from the data without using the labels. Unlike classification algorithms, no classification is predicted which implies that there is no specific way to compare model performance in most unsupervised learning methods.[2]

2.6.3 Regression

Regression is a supervised machine learning algorithm. Regression is used to find trends in the data. Regression is used for the estimation of relationships between a dependent and one ore more independent variables[51]. The regression model predicts a quantitative answer to a given input. The answer is often a continuous number. For example, a regression model could predict the distance travelled by a car given the speed and the time taken to travel the distance.

2.6.4 Classification

Classification falls under the category of supervised machine learning algorithms. The classification model when trained with labeled data predicts an output class for new unlabeled input data. The model performs the task of classifying an unlabeled input data into a labeled data. A simple example could be to classify the image as dog or cat given some input image. The below section distinguishes between three type of classification problems.

1. **Binary classification :** The task of classifying the input data into one of the two output classes is referred to as binary classification.

2. **Multi-class classification :** A Multi-class classification algorithm performs the

task of classifying the input data into one of the given n number of classes. The number of classes in this case is more than two.

3. **Multi-label classification :** Multi-label classification algorithm performs the task of predicting multiple mutually non-exclusive classes [10].

2.6.5 Dataset

A dataset can be simply defined as a collection of multiple data points. In case of a supervised machine learning algorithms, a dataset consists of data followed by the respective class label of the data. The complete dataset can be divided into three sections for the purpose of training mainly: A training set, a validation set, and a test set.

Training set : The training set is used to train the model. This data is passed to the supervised machine learning algorithm which extracts higher level and then eventually lower level features.

Validation set : The validation set is used to aid the model in computing the number of hidden parameters.

Test set : The test set is used to evaluate the selected model on unseen data.

2.6.6 Overfitting

Overfitting is a concept in the field of statistics and data science, which occurs when a statistical model fits exactly against the training data supplied to the model [22]. Whenever overfitting occurs, the model outputs inaccurate predictions against unseen input data. It is a crucial step to generalize a machine learning approach which learns features from new data and predicts an output. This is illustrated in the below Figure 2.3.

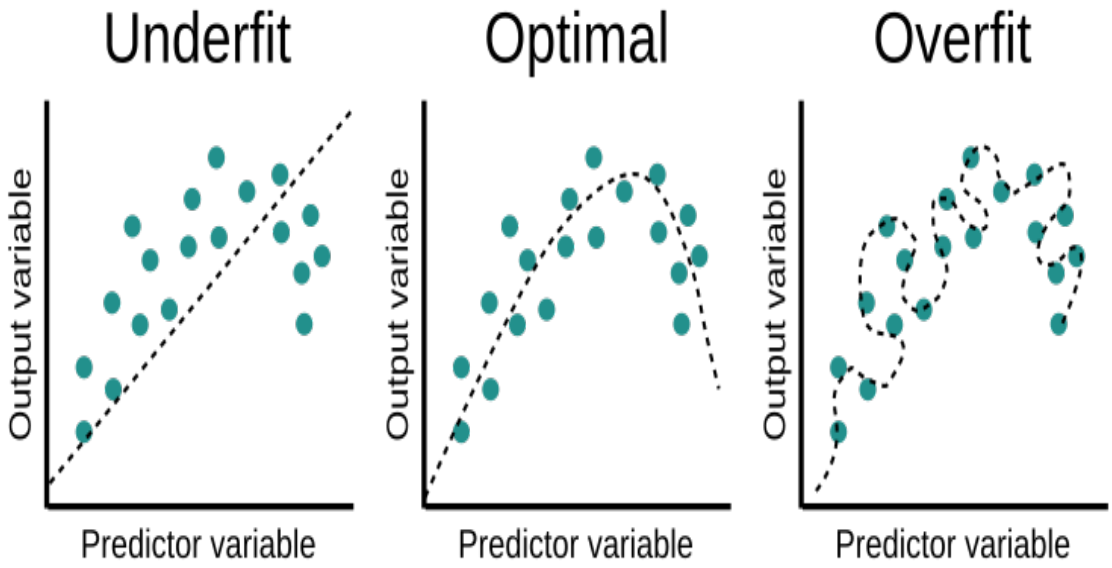

Figure 2.3: The dotted line shows the underlying function. The leftmost image illustrates underfitting, the middle image shows a good fit and the rightmost image illustrates the scenario of overfitting. Reprinted from [46]

2.6.7 Principal Component Analysis

Principal component analysis (PCA) is a dimensionality reduction technique. PCA is generally used to compress the dimensionality of large datasets. This could be achieved in two ways. The first approach is feature extraction where some lesser significant variables are eliminated from the data. The second approach is known as feature extraction. In case of feature extraction ,n new variables are created where n is equal to the number of original features in the dataset. After this, each of this new variable is represented as the combination of the old variables. PCA does the task of combining the input variables in an efficient way such that it drops the least important variables but still keeping the most valuable variables [8].

2.7 Deep Learning

Deep learning simulates the behavior of a human brain to perform classification and clustering with remarkable accuracy. Deep learning is the driving force behind

majority of the applications based on Artificial Intelligence. One of the practical applications of deep learning is artificial neural networks. A neural network architecture consists of three or more than three hidden layers which are responsible for finding out mapping between the provided input and output.

2.7.1 Neural Networks

Neural Network is the study of the internal architecture of the human brain in-order to induce intelligence artificially on machines [21]. An artificial neural network architecture consists of an input layer, one or more hidden layers, and an output layer. Each node is connected to another and it has a certain weight and threshold associated with it. The node of a neural network activates when its output is above a certain threshold value. The out put of this layer is passed to the next layer. Or else, no data is sent to the next layer of the network [21]. In the Figure 2.4, an architecture of a neural network can be seen.

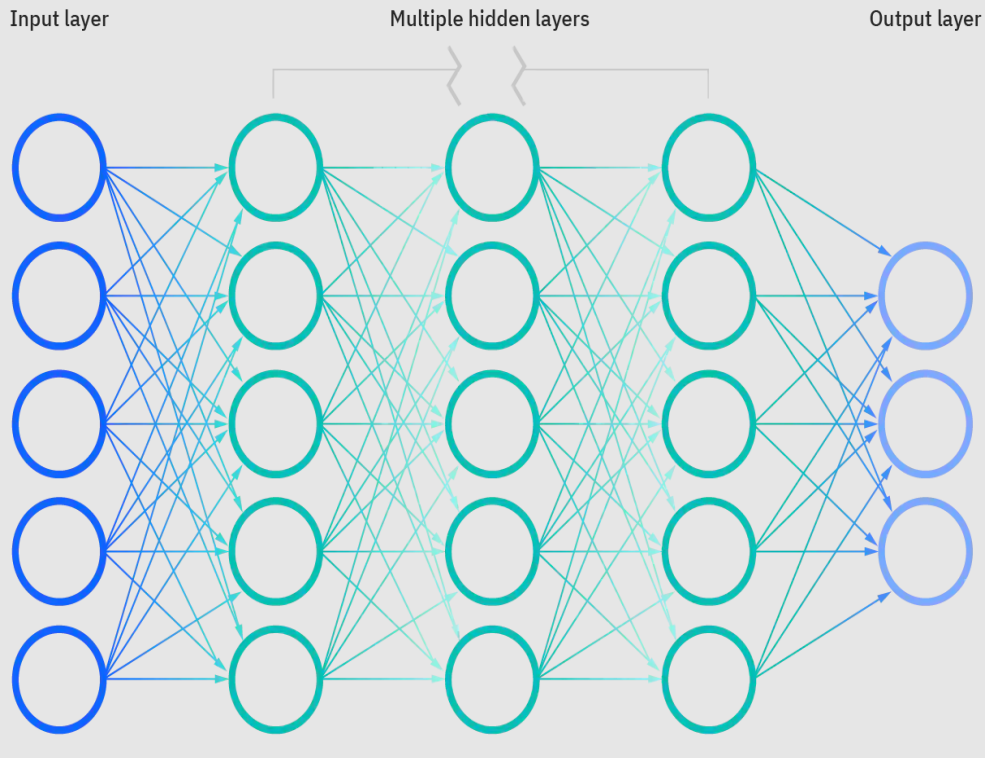

Figure 2.4: Neural Network architecture. Reprinted from [21]

The accuracy of neural networks tend to increase with huge amount of training data. However, it is the fine tuning of the various hyperparameters which enables us to improve the accuracy of the these networks allowing us to perform various tasks like classification and clustering at real time. Each individual node can be considered as a statistical function like regression, composed of some input data, weights, a bias (threshold) and an output. The formula would look like below:

$$\sum_{n=1}^{m} W_i X_i + bias = W_1 X_1 + W_2 X_2 + W_3 X_3 + bias$$

$$\text{Activation Function} = f(x) = \begin{cases} 1 & \text{if } \sum W_1 X_1 + b >= 0 \\ 0 & \text{if } \sum W_1 X_1 + b < 0 \end{cases}$$

Once an input layer is decided, certain weights are assigned to it. These weights aid the network in determining the most significant variables. A convolution operation is performed and the output of which is sent to the next layer provided it exceeds a certain threshold value. In case of supervised machine learning algorithms, the accuracy of a model is evaluated using cost (loss) function as below:

$$\text{Cost Function} = MSE = \frac{1}{m} \sum_{n=1}^{m} (y - \hat{y})^2$$

where,

i - index value of the sample

y - ground truth label

y-hat - predicted label

m - number of samples

2.7.2 Gradient Descent

The ultimate goal is to minimize the value of our cost function to ensure correctness of fit for test data input. The model adjusts its weights so as to minimize the value of the cost function. In order to minimize the value of cost function, the model adjusts

its weights and is also referred to as gradient descent. In the Figure 2.5, the point of convergence can be seen where the cost function is at its minimum.

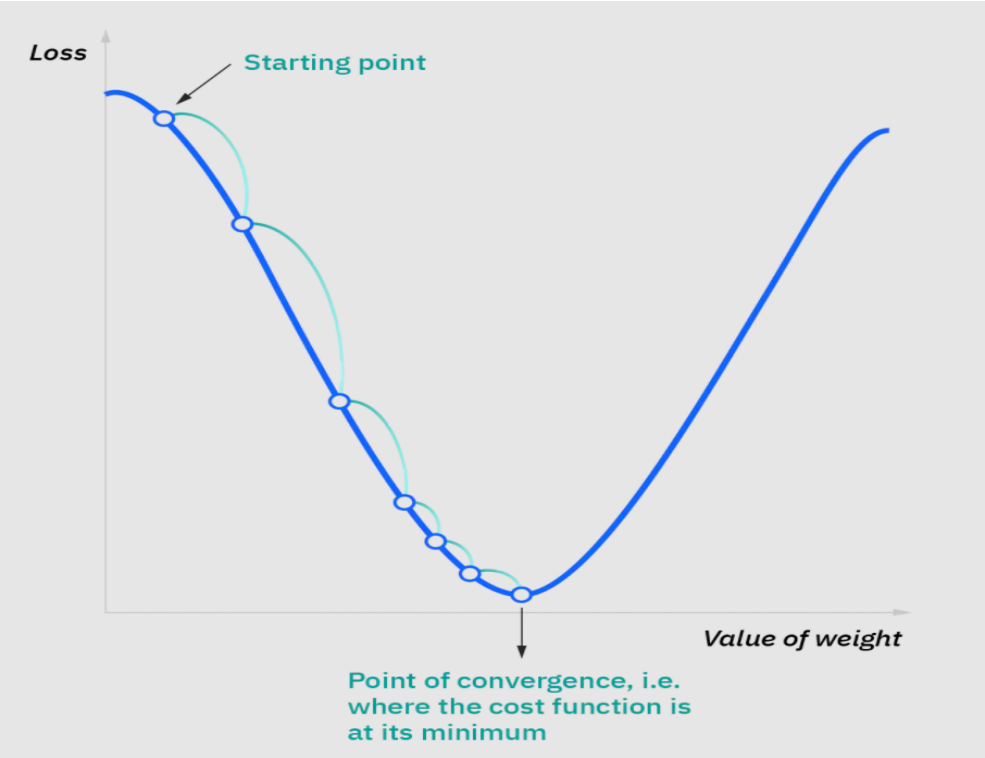

Figure 2.5: Gradient Descent. Reprinted from [21]

2.7.3 Convolutional Neural Networks

A convolutional neural network mainly consist of below layers:

- Convolutional layer
- Pooling Layer
- Fully Connected Layer

The convolutional layer is the core building block of a convolutional neural network. Input data and feature maps are fundamental components required by a Convolutional Neural Network[20]. The input image consist of 3 dimensions width, height and depth- which correspond to RGB in an image. A dot product is calculated

by applying the filter to an area of an image. The dot product is then fed into an array of output. The filter repeats this process by sliding over the entire image and calculates a dot product for each operation. The results of consecutive pooling operations is known as feature map.

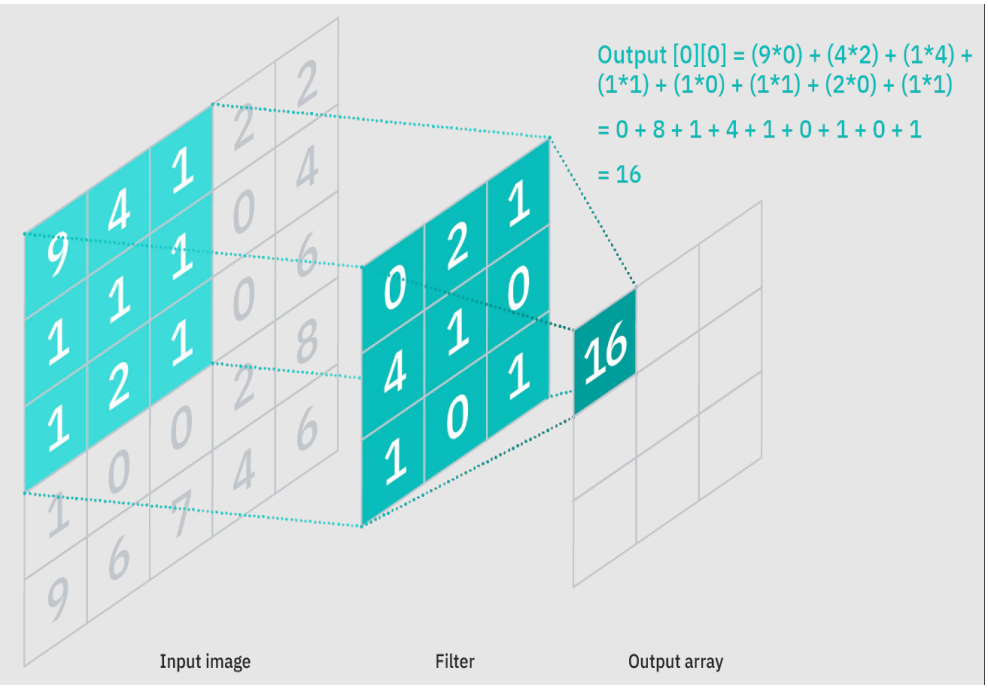

Figure 2.6: .Convolutional Neural Network

2.7.4 Transfer Learning

In case of transfer learning, a deep learning model is trained using a large and a reliable dataset and the model and its weights are saved which could be later used for a similar task with further training the model on some additional data. In case of transfer learning, the weights and biases learned by some other model can be reused for a similar task. In the Figure 2.7 it cab be seen how transfer learning benefits the training of a deep learning model.

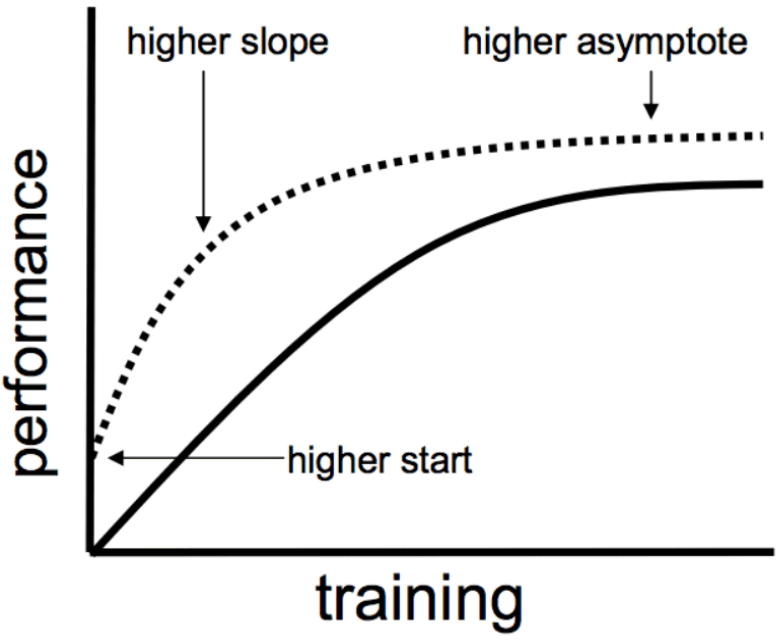

Figure 2.7: Transfer Learning. Reprinted from [60]

2.7.5 Pre-computed features

Pre-computing is the process of storing the output of all the layers except the last layer so that it could be used for some other task on the same dataset. It enables faster training of a network as the network is not trained from scratch. Since only the last layers change during the training and not the initial layers, input image data can be converted into these feature vectors instead of training the network using raw input images which saves us huge amount of computing power. In our case, pre-computed features extracted from ResNet-152 will be used [35].

2.7.6 3D Convolution Operation

In the context of action spotting, the task is not only to detect and classify an event but also capture the timestamp at which an event has occurred. Our network should be capable of finding out spatial as well as temporal information. Both our objectives can be achieved by using 3D convolution networks that are able to learn spatial as well as temporal information. Figure 2.8 illustrates the 3D convolution operation.

A video clip can be considered of the dimension C X T X H X W where, C - Number of channels (RGB), T - Number of frames in a video, H - Height in pixels, W -Width in pixels.

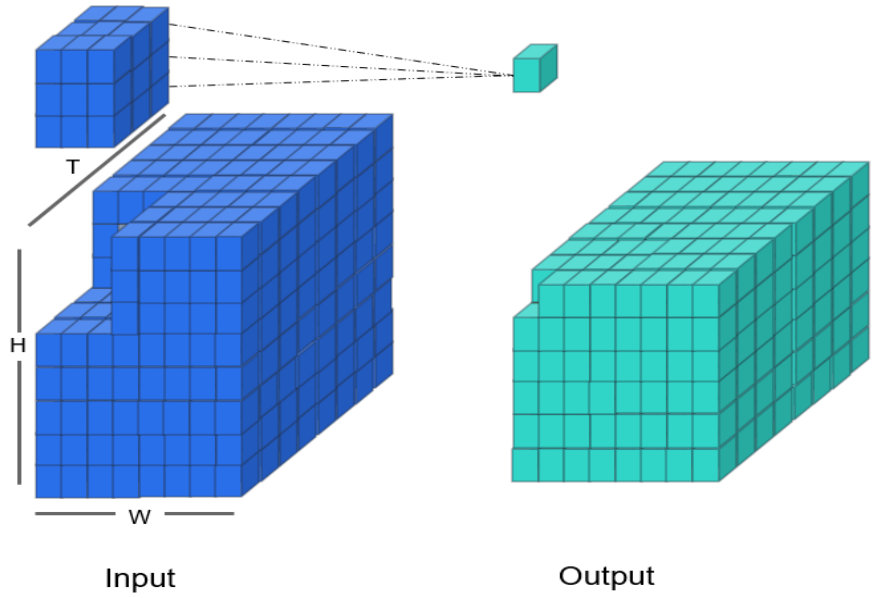

Figure 2.8: Illustrates 3D convolution operation where T = number of frames, W = width, H = height

2.7.7 Residual Connections

When deeper networks start converging, a degradation problem has been exposed: as the depth of the network increases, accuracy gets saturated and then the network degrades rapidly[35]. Traditional convolutional neural network proposed an idea of stacking a set of convolutional, pooling and an activation layer on top of each other. However, it was observed that adding more layers to a suitably deep learning model led to higher training error[35]. The degradation problem indicated that not all deep learning models are similarly easy to optimize. In case of traditional convolutional neural networks, there exists a direct mapping x -> y with a function H(x) between the input x and output y (a set of stacked non-linear layers). However, in case of

residual connections, a residual function can be defined using F(X) = H(x) - x which can also be rewritten as H(x) = F(x) + x , where F(x) and x represents the stacked non-linear layers and the identity function respectively. It is easier to optimize the function F(x) = 0 rather than F(x) = x using a stack of non-linear functions. F(x) is called residual function[35]. Comparison between a plain block and a residual block is illustrated in Fig 2.9.

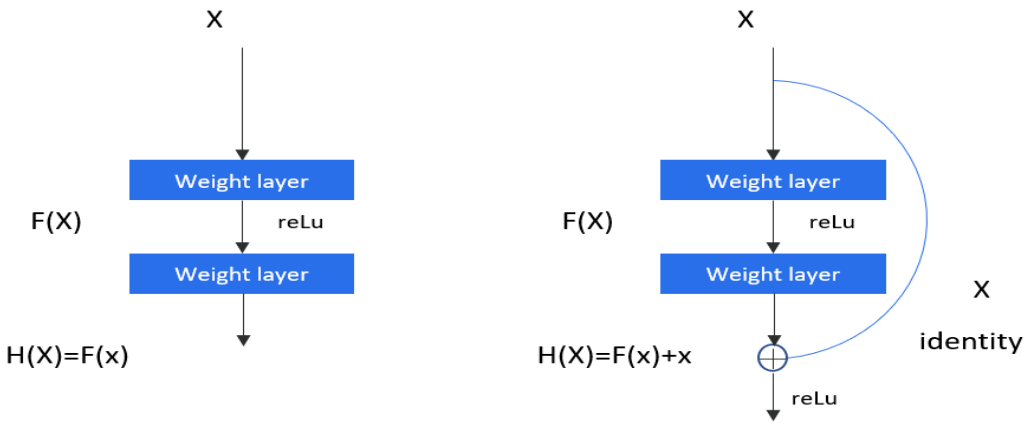

Figure 2.9: The image on left hand side illustrates the mapping between input and output of a CNN and the image on right hand side illustrates the mapping between input and output of a residual block

2.7.8 ResNet

Residual Networks, also known as ResNet is a neural network used as a backbone for many computer vision tasks. Residual block is a basic building block of a ResNet architecture. The fundamental breakthrough with the ResNet architectures was it allowed us to train extreme deep neural networks successfully[36]. The ResNet architectures consists of multiple residual blocks stacked side by side which makes the network wider instead of deeper and makes it comparatively easier to optimize than the deep CNNs [36]. In the below Figure2.10, an illustration of a residual block can be seen.

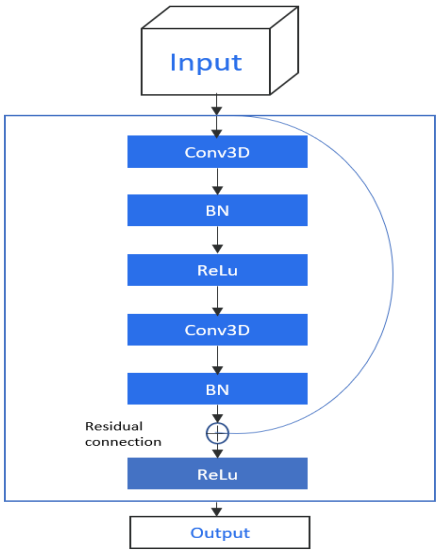

Figure 2.10: 3D Residual Block

,

2.7.9 Pooling

In case of Convolutional neural networks, a major problem with the output feature maps is that they are sensitive to the location of the features in the input. Poling layers in a CNN architecture provide an approach to down sample the feature maps [9]. The pooling operation slides over the input and then applies either maximum or average pooling to take out the maximum or the average value from the input feature map respectively. In the Figure 2.11 below, illustration of pooling operation can be seen.

2.8 Actionness

Actionness can be used as a unit of measure to quantify the likelihood of containing a generic action instance at a specific location. Wang et al. [70] proposed a novel approach to perform actionness estimation and further demonstrated how to apply actionness maps for action proposal generation and action detection. Figure shows 2.12 the actionness maps and actionness proposals for an image.

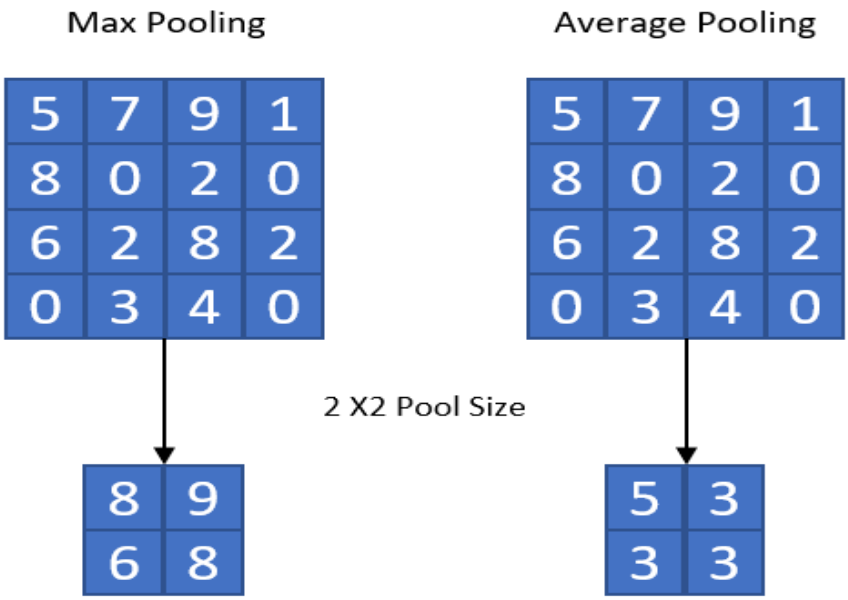

Figure 2.11: Leftmost image illustrates max polling operation and the rightmost image illustrates average pooling operation

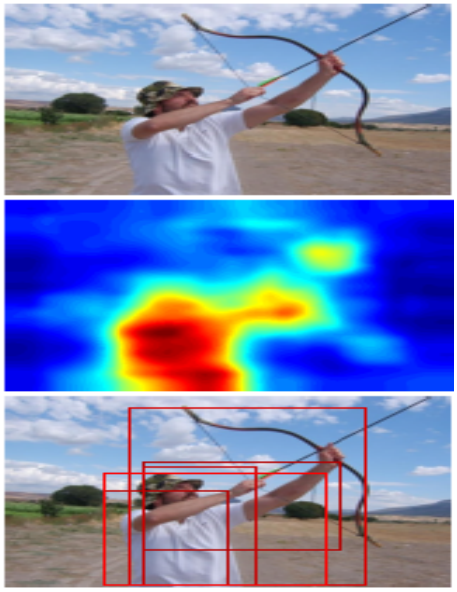

Figure 2.12: Actionness maps and proposals. Reprinted from [70]

2.8.1 Non maximum suppression

Most object detection algorithms use Non maximum suppression to trim down the number of detected bounding boxes to only a few. Most algorithms use a windowing technique in which thousands of windows of various sizes and shapes are created. To obtain probability of each class, a classification algorithm is applied. It is necessary to filter out the best predictions with the bounding boxes. NMS is the widely used algorithm for this task [48]. An example of of how NMS works is illustrated in the Figure 2.13.

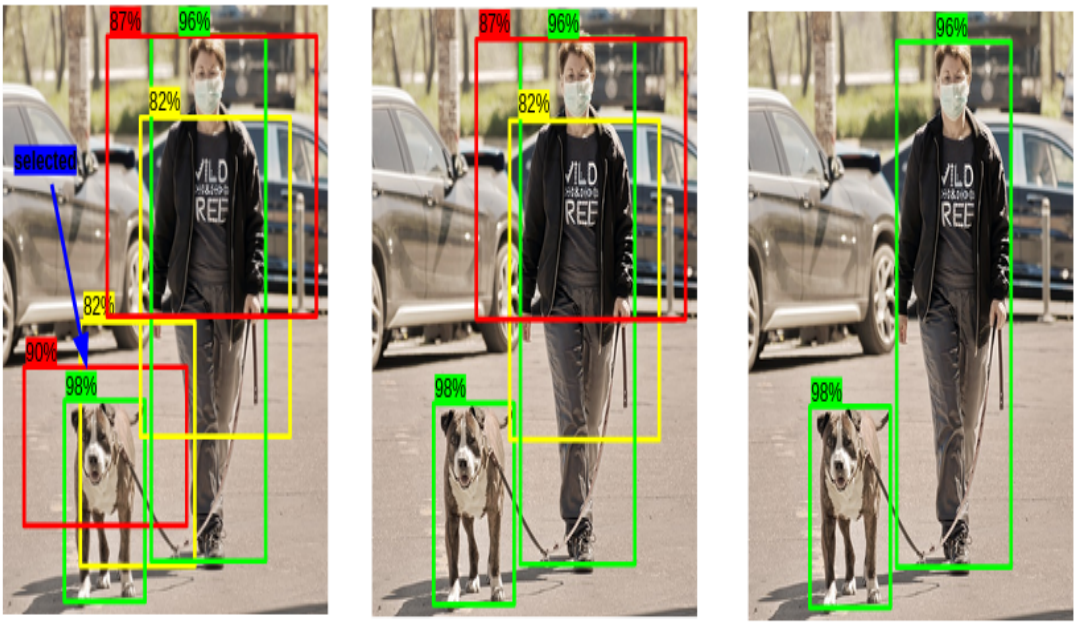

Figure 2.13: Non Maximum Suppression. Reprinted from [53]

2.8.2 VLAD

Jégou et al. [39] proposed an efficient approach of of searching the most similar images in a large image database. They proposed a descriptor, derived from both BOF and Fisher kernel [47], that aggregates SIFT descriptors and produces a compact image representation which was later coined as vector of locally aggregated descriptors (VLAD). Principal component analysis can be performed to reduce the

dimensions of VLAD without hindering the accuracy. VLAD is derived by extracting a region from an image and defining it using the 128-D SIFT descriptor. Clustering approach is used to assign the closest center of vocabulary size k to the descriptor. The size of k is set to 64 or 256. The vector differences between descriptors and cluster centers are assembled into a single k × 128 dimensional descriptor which is known as VLAD [6].

2.8.3 Datasets

Majority of the deep learning algorithms require large and scalable datasets for training. In order to perform tasks like action recognition and action localization, one requires high-quality annotated data. It is a tedious and an expensive process to annotate the data manually. During this thesis, pre-computed input features are used as input data to feed to the network. However, it is a very complicated process to manually annotate events in case of sports videos. It is beneficial to have such large corpus of annotated high-quality datasets, which are considered as benchmark datasets for various tasks in the field of video analysis.

Some of the early works that were used for the task of action recognition included UCF101[58] and HMDB-51 [42]. Plethora of research has been done in the field of action recognition recently and many different benchmark datasets have been released over the past few years. Some of these datasets include Sports-1M [40], YouTube-8M [3]. Another benchmark dataset called Kinetics-400 was initially released which consisted 400 different classes. Later, two more extended versions of the same dataset Kinetics-600 [13] and Kinetics-700 [14] were released with 600 and 700 different classes respectively. All of the above discussed became popular with the tasks like action recognition and action localization because of the temporal annotations provided in the dataset. Two more datasets THUMOS [38], ActivityNet [24] and the challenges proposed by the authors became quite popular with the community.

In soccer, a new dataset SoccerNet[32] was originally released in the year 2018. Later, an extended version of this dataset called SoccerNet-v2 [17] was released in the year 2020. This dataset comprises of 17 different classes with 300k temporally

annotations within SoccerNet's 500 untrimmed broadcast soccer videos. The authors of the SoccerNet-v2 [17] also presented some baseline deep learning approaches to test the dataset. The authors have also provided a set of pre-computed higher level semantic features obtained from various action recognition models. The authors have also provided reproducible code for some of the benchmark context aware deep learning approaches.

2.9 Related works

There exists an extensive amount of literature on action recognition. In the section below, some of the most recently published related work has been discussed. Understanding a video is one of the challenging tasks in the field of computer vision and numerous researchers have contributed to this field in the last few years. Action recognition has been identified as one of the tasks which is used to classify short and trimmed clips of a video. Considering the recent advancements in the field of artificial intelligence, a deep learning model could be used to perform action recognition. As majority of deep learning techniques require large number of data in order to train the model, many efforts have been dedicated toward collection of large-scale datasets. UCF101[58], HMDB[42] and YouTube-8m[3] datasets are used as benchmark to measure performance in the field of action recognition and have made the action recognition task more accessible.

Histogram of Gradients (HOG), Histogram of Flow (HOF), Motion Boundary Histogram (MBH)[16], dense trajectories[69], [68] dominated the field of video processing. In 2014, Karpathy et al.[40] studied various approaches to include the temporal information while training a Convolutional Neural Network to speed up the training process. They further studied the generalization performance of the best designed model by retraining the top layers on the UCF101[58] dataset and observed significant increase in the performance. Furthermore, Simonyan et al.[55] introduced a Two-Stream Convolutional Neural Network (ConvNet) architecture, which incorporates the spatial and the temporal networks. The architecture adheres to the two-stream hypothesis[33]. The Two-Stream network mimics the functioning

of a human visual cortex which contains two pathways: the ventral and the dorsal stream which perform the tasks of object recognition and motion recognition respectively. Extending this work further, Carreira et al. [12] introduced a novel two-stream inflated 3D (ConvNet). The filters and pooling kernels of the architecture were expanded into 3D, making it possible to learn spatio-temporal features. Feichtenhofer et al.[27] proposed a novel approach with 3D convolution and Pooling. Having seen the strong performance of the Two-stream Convolutional Networks, the Two-Stream architecture was further extended with ST-ResNet[26], which added residual connections[36] in the spatial and temporal streams. This approach was found to steadily increase the spatio-temporal receptive field as the network grew deeper.

Wang et al.[54][71] proposed Temporal Segment Networks (TSN) proposed an approach to train an action recognition model with limited number of training data. TSN extracts short snippets / samples over a long video, where the samples distribute uniformly along the temporal dimension. The final prediction is then made by fusing the predictions from samples. Tran et al.[62] devised a new class of deep learning model called C3D which was capable of learning spatial as well as temporal features from the input. The Two-Stream Inflated 3D ConvNet (I3D) [12] model was introduced along with the kinetics-400 [41] dataset. The reason this two-stream architecture was called inflated is because the model possessed two streams (one temporal and one spatial), and the spatial stream was pre-trained on the ImageNet[18] dataset and further the 3D filters were inflated so that they could be used for performing the 3D convolution operation.

To avoid relying on only RGB or optical flow as input, PoTion used a pose detection algorithm [11], where an approach was proposed to detect the 2D pose of multiple people inside an image. This algorithm achieved good results irrespective of the number of people in the image. However, this approach was overly reliant on the pose detection model results. Later, Tran et al.[61] introduced Res(2+1)D, which separates the 3D convolution into 3 steps. It was easier for the network to model the spatial and temporal context separately. Finally, Feichtenhofer [28] introduced the SlowFast architecture.

THUMOS [38] challenge was introduced to serve as a benchmark for the task

of action recognition. Prior to the THUMOS [38] challenege, action recognition, including the THUMOS [38] challenge had mainly focused on the classification of trimmed videos. To address the challenge of action recognition in untrimmed videos, Shou et al.[54] proposed an approach based on three segment-based 3D ConvNets. First, a proposal network identifies candidate segments in a long video that may contain actions, a classification network to solve the one-vs-all action classification problem and finally a localization network which fine-tunes the learned classification network to localize each actions instance. They also proposed a novel loss function to discover temporal overlap and achieve high localization accuracy.

Ekin et al. [23] introduced a novel framework for analysis and summarization of the soccer videos. Tsagkatakis et al. [64] devised an approach to independently encode the spatial and the temporal features and fuse their features using Autoencoders. They achieved high classification accuracy with limited number of data samples, by leveraging pre-trained models with the fine-tuned fusion of the spatial and the temporal features. Qiu et al.[49] proposed a new architecture named Pseudo-3D Residual Net (P3D ResNet), by simulating 3 X 3 X 3 convolutions with 1 X 3 X 3 convolutional filters on spatial domain plus 3 X 1 X 1 convolutions to construct temporal connections on adjacent feature maps in time. They achieved clear improvements on the classification task of Sports-1M [40] dataset. Zolfaghari et al. [75] put forward a network architecture which takes long-term content into account and enables fast per-video processing at the same time. Tran et al. [63] devised an approach to factorize 3D convolution operation into spatial and temporal interactions. The authors believed that it increased the accuracy of the network whilst saving training time. Feichtenhofer[25] presented the X3D network which produced excellent results on the 2D image classification. X3D achieved state-of-the art performance with 4.8 and 5.5 times fewer multiply-additions and parameters for similar accuracy than the previous work [25]. Wu et al. [72] proposed a multigrid method for efficiently training video models. Training competitive deep video models are slower than training their counterpart image models. The authors presented an approach to use variable mini-batch sizes with different spatio-temporal resolutions that are varied as per the schedule. They empirically demonstrated a general and robust grid schedule and tested this

approach on different models (I3D, non-local, SlowFast) and achieved comparable results to state-of-the-art models on benchmark datasets like Kinetics,Charades,etc without any loss in accuracy. Brattoli et al. [7] presented an approach called Zero-shot learning (ZSL) which trains the model once and generalizes well to new tasks whose classes are not present in the original dataset. They used trainable 3D CNN architecture to extract the visual features and achieved comparable results to state-of-the art architectures.

The task of Action spotting was introduced in the SoccerNet[32] dataset. Action spotting is the task of predicting a class and a point in time at which an event has occurred. A benchmark dataset called SoccerNet[32] was introduced to make the task of action spotting easier. The dataset is composed of 500 complete soccer games from six major European leagues, spanning over a total duration of 764 hours. The authors devised a baseline model which recorded a Average-mAP of 49.7%.

Cioppa et al.[15] proposed a novel Context-Aware loss function (CALF) . For the task of action spotting, the loss function takes into account the set of events occur surrounding an event. They evaluated their model against the benchmark SoccerNet dataset and achieved an overall Average-mAP of 62.5%. The authors evaluated the approach against the ActivityNet [24] dataset. Vats et al.[66] introduced a multi-tower temporal convolutional network architecture for the task of action spotting. They achieved an Average-mAP of 60.1%. Vanderplaetse el al. [65] introduced a multi modal approach which takes into consideration the audio as well video information for the task of action spotting. Action spotting is the task of finding out the temporal anchors of the video events and classifying them. They evaluated their neural network model against the benchmark SoccerNet dataset and observed a Average-mAP score of 56.0% for the task of action spotting. Hurault et al. [37] proposed a self-supervised pipeline which could detect and track soccer players in low-resolution irrespective of the video recording conditions and without using any manually annotated labeled data. Their model achieved competitive results in comparison with the state-of-the architectures.

In the year 2020, Deliège et al. [17] proposed SoccerNet-v2, a novel large-scale collection of manual annotations of the original SoccerNet [32] dataset. They released

around 300k annotations within the SoccerNet's 500 untrimmed soccer videos. They introduced multiple challenges like action spotting, replay grounding and camera shot segmentation,etc. along with reproducible benchmark results for each task. Rongved et al. [45] presented a novel approach to tackle the task of action spotting using 3D convolution models. Their goal was to create a model which would not only yield accurate results but also can be used in a real-time setting. They evaluated the model on three different datasets from SoccerNet [32], the Swedish Allsvenskan, and the Norwegian Eliteserien. They achieved a relatively lower Average-mAP score of 32.0% on a small temporal window of 8 seconds.

Tomei et al.[59] proposed a novel approach for the task of action spotting which could simultaneously predict the event label and its temporal offset using the same underlying features. They proposed two novel strategies to enrich the training of the deep learning model: the first one for data balancing and the second one for masking ambiguous frames with randomly sampled background frames. When evaluated against the benchmark SoccerNet dataset, they achieved competitive results to state-of-the art models. Additionally, they fine-tuned the model with a strong 2D convolutional backbone for feature extraction and further achieved Average-mAP score of 75.1%. Giancola et al. [31] presented a novel pooling method to model the temporal semantics in a soccer broadcast video for the task of action spotting. Unlike the other pooling methods which consider the temporal information to pool from, they split the context into two parts: context before the occurrence of action and context after the action has occurred. They introduced a novel pooling method called as NetVLAD which was able to learn the context in a feature space using a clustering algorithm based approach. They trained and evaluated their novel pooling method against the large -scale SoccerNet-v2 [17] dataset and reached Average-mAP score of 53.4%. Mahaseni et al.[43] proposed an approach considering the long-range dependencies between video frames for accurate event localization. The two-stream CNN extracts spatio-temporal features and the dilated recurrent neural network utilizes this information for the task of action spotting. After evaluating their model on the SoccerNet [32] dataset, they observed Average-mAP score of 63.3%. Zhou et al.[74] further presented an approach in which they released pre-computed features

in which they concatenated the output features extracted from 5 different action recognition models along the feature dimension.

2.10 Summary

In this section, all the relevant concepts and terminology required to understand the thesis was presented. It included all the fundamental concepts with respect to machine learning, deep learning and neural networks. The above section 2.9 presented the extensive amount of research that has been done in the past few years and also presented their results with respect to different performance metrics. From the above research it was discovered that automatic detection of events in sports videos is an active area of research. Further, the benchmark datasets used for the task of action recognition and action spotting were discussed, mainly the SoccerNet-v2 [17] dataset.

Multiple different approaches that have been used in the past over the last few years have been presented. These approaches were mainly focused on using machine learning and deep learning techniques for the task of action recognition. When it comes to the task of action spotting, it becomes comparatively difficult to model the temporal context. Some of the early works include two-stream convolutional neural network architectures introduced by Simonyan et al.[55] which incorporated the spatial and temporal networks produced promising results. Further, Tran et al.[62] introduced a family of ResNet 3D architectures which separated the 3D convolution into two steps which was able to model the spatial and temporal context separately.

In 2020, an extended version of SoccerNet [32] dataset called SoccerNet-v2 was released. Giancola et al. [31] presented deep learning approaches which were able to model the temporal context in soccer broadcast videos using a clustering algorithm based approach for pooling. Giancola et al [31] evaluated their deep learning approach against the SoccerNet-v2 dataset and yielded comparable results with the state-of-the-art architectures. Cioppa et al.[15] introduced a contextual loss function which further improved the results on the task of action spotting. Zhou et al. [74] presented an approach where the authors fine-tuned multiple action recognition

models to extract high-level semantic features and also designed a transformer based temporal detection module to locate the target events.

The above research shows that automatic detection of events is still an active area of research. Considering most of the results presented in the above section yielded promising results, not all of them can be used in real time setting considering it needs to be highly accurate. After reviewing that most of the approaches use single model features for this task, our experiments were conducted with concatenated features[74].

Chapter 3

Methodology

For soccer events, a summarized clip comprises of sports events such as goals, bookings, offside, penalties, etc. At present, the golden standard to create such summarized events, the sports events are manually tagged by human operators. Manual annotation of events consumes plenty of time and is a tedious process. Automating this process would save huge amount of time and efforts. With the use of deep learning approaches, automatic detection of sports events cane be performed. Extensive amount of research has taken place in this field of video analysis for the tasks such as action recognition and action localization. Action recognition and localization is still an active area of research and various datasets have been released over the years which are considered as benchmark datasets for the task of action recognition. During this thesis, the deep learning approaches will be evaluated against the benchmark SoccerNet-v2 [17] dataset. In addition, the experiments were performed by using pre-computed features concatenated from 5 different 3D convolution models as feature extractors.

In this section 3, the dataset is described in-detail along with the 17 classes and the temporal annotations. In addition, it is further discussed how the pre-computed concatenated features using the 3D convolution models as feature extractors were extracted. Furthermore, the impact of using concatenated features and the hyperparameter setting for modelling the temporal context is discussed. Also, the implementation details about the contextually aware deep learning were

provided. In the end, various evaluation metrics have been discussed to asses the performance of the models.

3.1 Dataset description

The SoccerNet-v2 dataset presented by Giancola [17] was used for evaluating our deep learning approaches. This dataset is majorly used for the task of action spotting. The SoccerNet-v2 dataset is not only used to classify the events for action recognition but also for temporal localization of events. The dataset consists of 550 games from the major European soccer leagues. The dataset includes the data for three seasons from 2014-2017. Table 3.1 shows the distribution of league games spanned across multiple seasons. It includes the data from the major footballing European leagues mainly: England (EPL), Spain (LaLiga), Germany (Bundesliga), France (Ligue 1), and Italy (Serie A), and the UEFA Champions League. The distribution of league games per league is illustrated in Table 3.1 below:

Field Name	**14/15**	**15/16**	**16/17**	**Total**
EPL	6	49	40	95
LaLiga	18	36	63	117
Ligue 1	1	3	34	38
Bundesliga	8	18	27	53
Seria A	11	9	76	96
UCL	37	45	19	101
Total	81	160	259	500

Table 3.1: Distribution of league games per league and season

The dataset is structured in a format where each folder contains two videos each for the respective half of the game in MKV format and contains 764 hours of untrimmed soccer broadcast videos. The authors of the dataset have also provided the ground truth label data temporally annotated with timestamps of events in JSON format. In addition to this, the authors have also provided pre-computed features by

using various 3D convolution models as feature extractors. The dataset is split into three parts: training, test and validation and each part covers 300,100 and 100 games respectively. The networks have been trained using pre-computed features instead of using the raw video clips to save computational complexity.

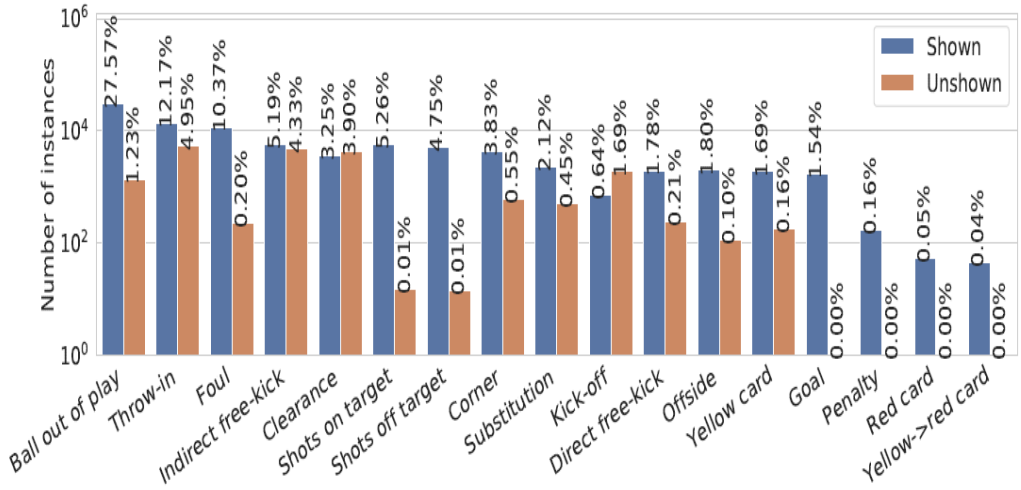

Figure 3.1: SoccerNet-v2 classes. Reprinted from SoccerNet-v2 [17]

The temporal annotations in SoccerNet-v2 are provided at the center of each frame[17]. This gives us the exact timestamp of event to be predicted. The dataset contains contains such 300000 temporally annotated events spanned across 500 soccer broadcast games. Figure 3.2 shows the 17 different classes included in the dataset and the Fig 3.1 illustrates the distribution of all the 17 classes in the dataset. The category shown and unshown represents whether the event was visible in the video clip or not.

Three different types of pre-computed features are provided along with the SoccerNet-v2 dataset: C3D [62], I3D [12] and ResNet [35]. The models are tested using ResNet-152 visual features. The ResNet-152 [35] model pre-trained on ImageNet [18] as an image classifier is used as a feature extractor. These features are extracted at 2FPS. Further, dimensionality reduction is performed using PCA to compress the dimension size to 512. The feature extractor pipeline is illustrated in Figure 3.3.

In addition to the pre-computed features provided by SoccerNet [32], pre-

Figure 3.2: SoccerNet-v2 classes

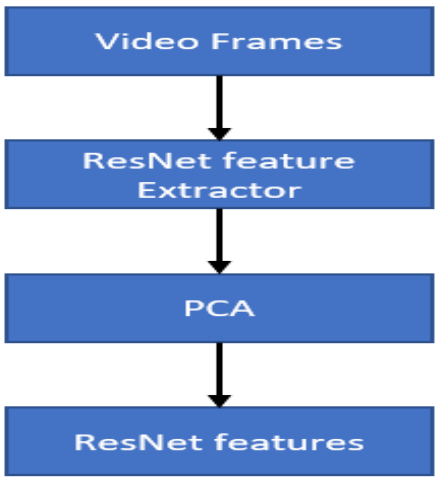

Figure 3.3: Feature extraction pipeline using ResNet model as backbone

computed concatenated features provided by Baidu [74] have been used. The selected contextually aware deep learning approaches were fine tuned by optimizing some of the hyperparameters of the model.

3.2 Model selection

The selected models should be fine tuned by optimizing the hyperparameters and this section describes the models that were best suited for this task. The models were further tested using concatenated pre-computed features from multiple models. Hence, the below models were selected as per our needs for performing the experiments:

- The model from Cioppa et al. [15] which proposed a novel loss function for modelling the context in videos called Context-Aware Loss Function (CALF).The context aware loss function was tested for the task of action spotting.
- The temporally aware pooling model from Giancola et al.[31], which uses a pooling method based on VLAD (vector of locally aggregated descriptors). The

temporally aware pooling method was tested for the task of action spotting.

- The above specified models are further tested with input as pre-computed features obtained by concatenating output features obtained by implementing the approach proposed by Zhou et al.[74].

3.2.1 CALF method

Reason for model selection

The task of action spotting was introduced in the paper SoccerNet[32] where the objective is retrieving the exact time when an event occurs in the given video frame. Cioppa et al.[15] proposed a pipeline where they considered natural context surrounding the actions and later they incorporated this knowledge into a novel loss function that performs some temporal segmentation. Further, they design an action spotting module to detect reactions based on this temporal information.

In order to observe difference in the performance of the models for action spotting on soccer videos, the model was tested using concatenated pre-computed input features from multiple 3D CNN models and pre-computed features from single model like ResNet-152. The models were selected in such a way that they are able to determine context in the given video clip. The Context Aware Loss Function (CALF) model was proposed by Cioppa et al. [15] and is illustrated in Figure 3.4.

The CALF model proposed by Cioppa et al. [15] was among the top 5 contextually-aware deep learning approaches and yielded comparable results to state-of-the-art for the task of action spotting. It was observed that the code was publicly available for the above specified deep learning models which made it feasible for us to fine-tune and use the baseline model for further training and testing our code.

It was observed that the CALF method yielded comparable results to state-of-the-art contextually aware deep learning approaches and as the code is publicly available to implement the baseline model. It further allowed us to analyze the pooling technique and the loss function used in this approach and perform further analysis on it.The model was further tested using concatenated features from multiple model as

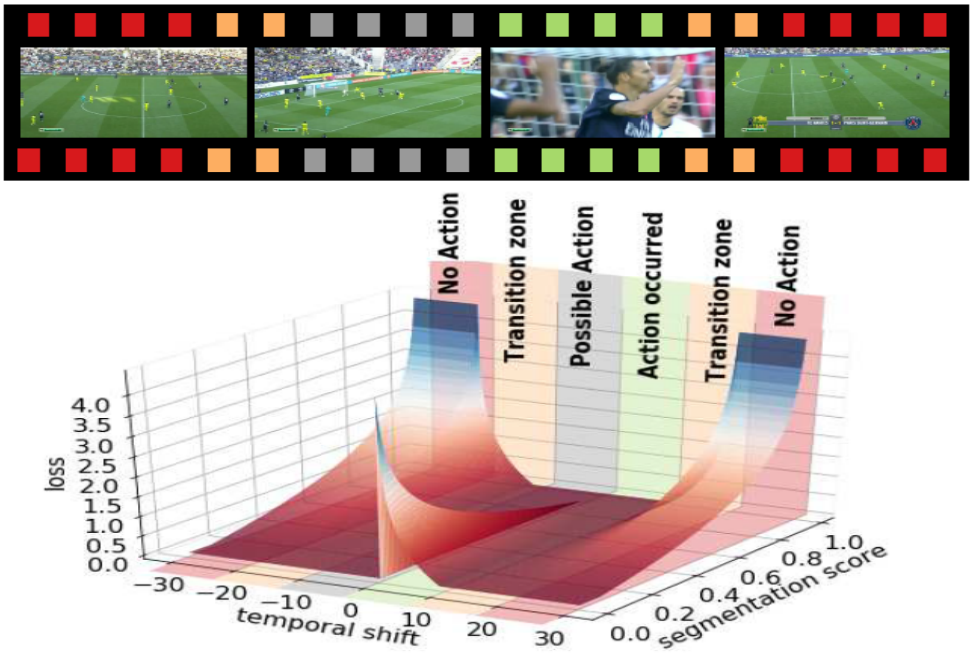

Figure 3.4: Contextually Aware Loss Function. Reprinted from Cioppa et al. [15]

input.

Description

The Context Aware Loss Function method also known as CALF was proposed by Cioppa et al.[15]. For the task of action spotting, Cioppa et al. [15] introduced a novel loss function. The function models the temporal context surrounding an event instead of keeping a single frame as center of attention. Their network consisted of a frame feature extractor module and a temporal CNN module which was responsible for outputting the class feature vectors for each frame, a segmentation module followed by an action spotting module. Figure3.5 illustrates the architecture of the loss function.

To perform the task of action spotting, Cioppa et al.[15] developed a context-aware loss for a temporal segmentation module, and a YOLO-like [50] loss for the action spotting module. Annotations for the segmentation module and spotting module are encoded. To train the network, the annotations are re-encoded in two different ways: first with a time-shift encoding for the temporal segmentation loss, and then for

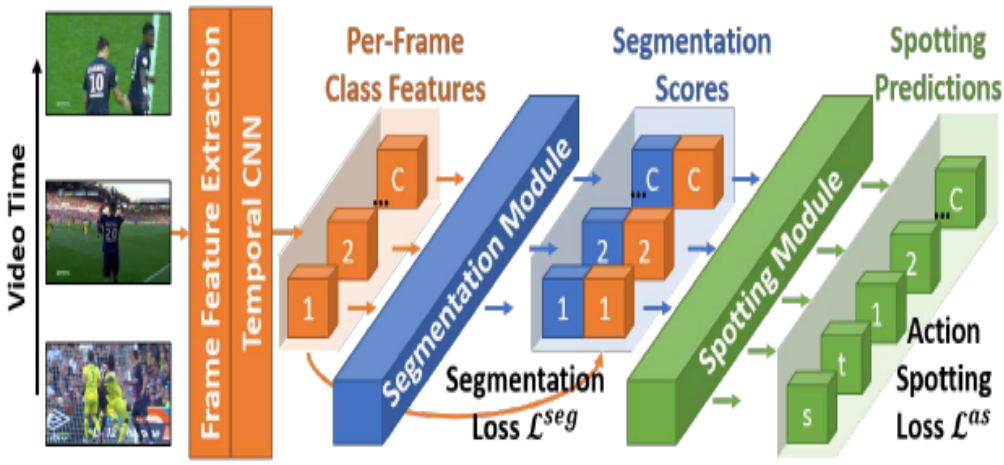

Figure 3.5: Pipeline of Contextually Aware Loss Function. Reprinted from Cioppa et al. [15]

computing the actual loss for the spotting module, YOLO-like encoding is performed [15].

In order to calculate the time-shift encoding, the temporal context around each action is split into multiple segments with respect to their distance from the action, as illustrated in Figure 3.6. Frames that are far before, just before, just after and far after action are grouped [15]. The segmentation scores are assigned by the temporal segmentation module based on below concepts:

1. Far before spotting an action of some class, its occurrence can not be anticipated.
2. Just before the action occurs, its occurrence is uncertain. Hence, the score for that class does not particularly point towards any direction.
3. Just after an action has happened, many visual cues suggest the detection of action. hence, the score for that class imparts the presence of an action.
4. Far after an action, the score for that class should impart that no action is occurring anymore. [15]

The loss function was designed in such a way that it leverages the temporal context around the action spot where the temporal shift is 0 as illustrated in Figure

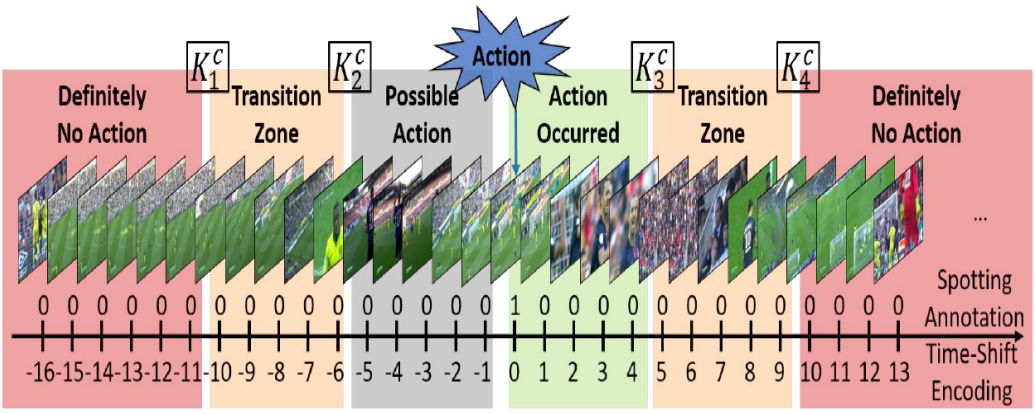

Figure 3.6: Time-shift encoding used in the Conext-Aware Loss Function which constitutes to action context slicing. Reprinted from Cioppa et al. []

3.4. All the frames except the frame at which action occurs have been assigned a spotting annotation as 0. In order to differentiate between the frames they have been assigned time-shift encoding as shown Figure 3.6 which makes the annotations richer. The action context slicing is delimited by context slicing parameters as shown in the Figure 3.6. The loss function is responsive to the positioning of the frames. Frames which are far off from the point of action are heavily penalized. Frames which are moderately closer to the occurring action are less penalized. Frames which are just before the action occurs are not penalized to avoid providing misleading information as it does not guarantee that an action has occurred. However, the frames which are present just after an action occurs are heavily penalized which indicates the occurrence of an action. the model is unsure of what action takes place before an annotated event occurs and hence the model does not heavily penalize the frames just before the action. Finally, the last two segments are the transition zones between the two nearby segments to the event, and the two distant segments.

The network takes video as input and outputs an actionness prediction score for each frame of the video. The model splits the videos into smaller chunks and for each chunk the model tries to spot the actions. The actions are encoded in a YOLO-like fashion [50] with a set of vectors one per action limited to five actions per chunk. In the vectors, they encoded following three elements: 1) The presence or absence of

actions 2) The locations within the chunk and 3) The class of the actions.

The ground truth vector encoding is illustrated in below Figure 3.7: The network

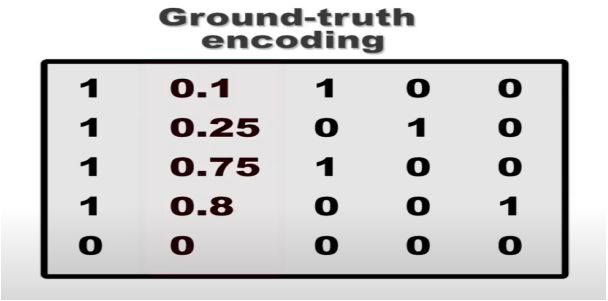

Figure 3.7: CALF ground truth encoding. Reprinted from [1]

predicts 5 action vectors per chunk as illustrated in Figure 3.8. They leveraged an iterative one-to-one matching to pair each prediction with a ground-truth vector based on which has the closest location in the chunk as illustrated in Figure 3.9. The action spotting loss between the predictions and ground-truth vectors is the computed as a weighted sum of squared errors. The spotting module uses the previous contextual features and detects the detected action spots.

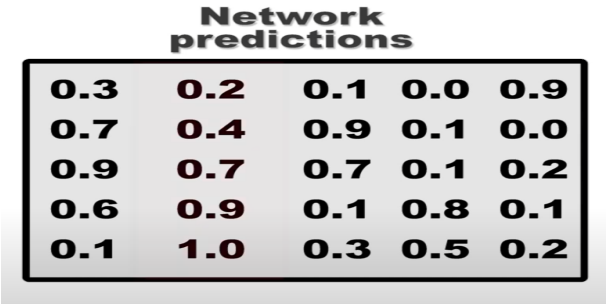

Figure 3.8: CALF Network predictions. Reprinted from [1]

Implementation

The baseline code supplied with the original model was used for training the CALF [15] network. In order to perform our experiments for different configurations, an Nvidia DGX-2 was used which comprises of 16 Nvidia Tesla V100 GPUs each having a RAM of 32 Gigabytes which totals to a capacity of 512 Gigabytes.

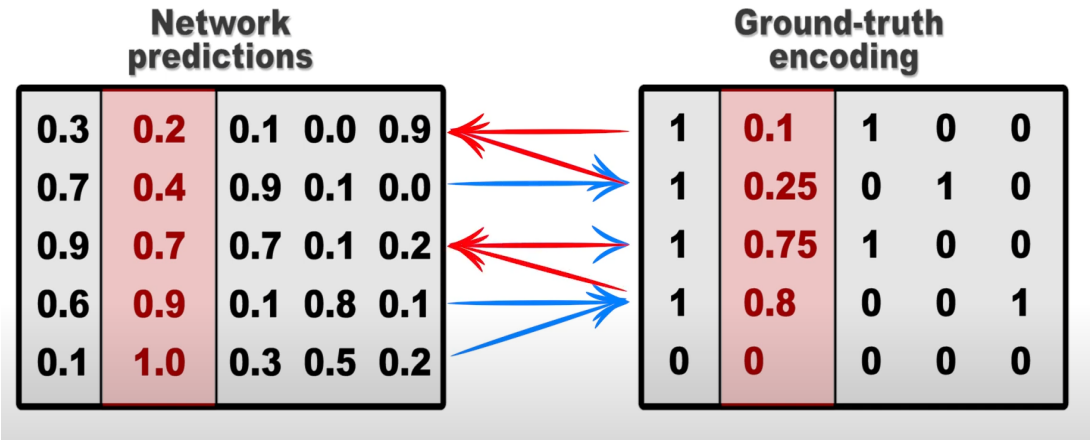

Figure 3.9: The spotting module performs iterative one-to-one matching between ground truth labels and the network predictions based on which has the closest location in the input chunk. Reprinted from [1]

The experiments were performed with ResNet-152 features and concatenated input features that combine the features concatenated from multiple 3D convolution models. The concatenated features were extracted by using an approach stated by Zhou et al. [74]. The model was validated after every 20 epochs after training the CALF method for 300 epochs. Non maximum suppression was applied to [48] the predictions to filter our weak actionness proposals.

3.2.2 Temporally Aware pooling Method

Reason for model selection

In order to perform automatic detection of events in sports videos, a major task consists in understanding the high-level semantic information in the sports videos. Giancola et al. [31] proposed an analysis on action spotting in soccer broadcast videos, which consisted of temporally localizing the important actions in a soccer videos. They proposed a novel feature pooling method based on NetVLAD[31]. They named the pooling method as NetVLAD++. The NetVLAD++pooling method was able to determine temporal context in a soccer broadcast video. Majority of the deep learning networks either use either max-pooling or average-pooling to pool over the features. However, Giancola et al.[31] designed a pooling method which was able to determine

the context before and after an action takes place. The NetVLAD pooling layer was able to separate the past and future context from the frames in a soccer broadcast video. The module then learns a set of vocabularies for the past and future context. The authors stated that introducing such prior knowledge aids the pooling layer in modelling the context in a better way[31].

NetVLAD++ [31] consists of two NetVLAD pooling layers one after and are responsible for pooling over the features in frames before and after the action occurs, respectively. Injecting such temporal awareness brings a significant boost in the performances in the SoccerNet-v2 benchmark. The NetVLAD++ pooling method [31] not only introduced a novel pooling method which that learns a temporally-aware vocabulary for past and future temporal context but also was efficient in in term of memory and computational complexity.

NetVLAD

VLAD [6] forms the basis of NetVLAD [31] pooling layer. NetVLAD [31] provides a generalized representation of VLAD.

VLAD learns codebook of k visual words computed with k-means clustering. The VLAD method accepts a set of N D-dimensional features as input, a set of k clusters centers with same dimension D as VLAD parameters, the output of the VLAD descriptor V [31] defined as below:

$$V(j,k) = \sum_{i=1}^{m} a_k(x_i)(x_i(j) - c_k(j))$$

where x_i(j) and c_k(j) are the j-th dimensions of the i-th descriptor and k-th cluster center respectively. $a_k(x_i)$ represents the hard assignment of the samples x_i from its closer center, i.e $a_k(x_i)$=1 if c_k is the closest center of x_i, otherwise it is assigned a value of 0. normalization is performed on the vector V and then it is flattened into a vector of dimension D X K.

To avoid hard assignment, the NetVLAD [31] pooling method introduced a novel soft assignment $\hat{a}_k(x_i)$ for the samples $(x_i)_{i=1}^{N}$, based on their distance from each

cluster center.

$$\hat{a}_k(x_i) = \frac{e^{-\alpha||x_i - c_k||^2}}{\sum_{k'=1}^{k} e^{-\alpha||x_i - c'_k||^2}}$$

The value of $\hat{a}_k(x_i)$ ranges between 0 and 1. The parameter α controls the softness of the assignment. The above equation can be considered as softmax layer of a convolutional layer for the input features parameterized by $w_k = 2\alpha_k$ and b_k = -$\alpha||c_k||^2$ [31]. Further, $\hat{a}_k(x_i)$ is defined as below:

$$\hat{a}_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b'_k}}$$

The values were substituted from the above equation into the VLAD formulation. The NetVLAD features are defined as in equation below. The original VLAD optimizes only the cluster centers c_k. NetVLAD method optimizes the values of w_k,b_k and c_k independently, where w_k= $2ac_k$ and b_k = -$a||c_k||^2$ [31].

$$\hat{a}_k(x_i) = \frac{e^{w_k^T x_i + b_k}}{\sum_{k'} e^{w_{k'}^T x_i + b'_k}}(x_i)(x_i(j) - c_k(j))$$

Provided an temporal window in a video clip, the NetVLAD method is capable finding out whether specific actions occur in the window or not. For inference purpose, NetVLAD provides class-wise actionness scores by a sliding window approach and further applies Non maximum suppression to filter out the actionness detections.

NetVLAD++

Giancola et al. [31] proposed a temporally aware pooling module called NetVLAD++. The VLAD [6] and NetVLAD [31] modules do not consider the order of the frames in a video, however treat the features as a set. For the task of action spotting, the frames from the video can be ordered temporally and can be semantically disentangled into past and future context.

Cioppa et al. [15] argued that the contextual information before and after an action occurs is different, yet it complements each other. In addition, Giancola et al.[31] further argued that disparate actions might share similar vocabulary either before or after an action occurs, but not both.For example, the semantic information obtained before a goal is scored and a shot on goal is taken could be similar, representing a lower level semantic concept of a player taking a shot on goal and a goalkeeper catching the ball. Yet, these two actions represent different contextual semantics after the action occurs, with the fans cheering when a goal is scored or express their disappointment when the shot attempt is blocked by the goalkeeper respectively. The amount of information to pool among the features before and after an action occurs might contain different low-level semantics, helping the module in identifying lower-level granular features. The architecture of NetVLAD and NetVLAD++ is illustrated in the Figure 3.10. NetVLAD++ [31] learns two different pooling modules for the frame features from the frames that occur before and an action occurs. The context before and after an action occurs is defined as $[-T_b, 0]$ and$[0, T_a]$ respectively. Each pooling module in NetVLAD++ amalgamates information from 2 subsets of features, using K_a and K_b clusters for after and before clusters respectively [31]. NetVLAD++ is defined as:

$$V = \omega(V_b, V_a)$$

where ω represents the aggregation of V_b and V_a that represent the NetVLAD pooled features for the frames before and action occurs respectively.

Giancola et al. [31] further integrated their pooling module into an architecture depicted in Figure 3.11. The architecture is based on a pre-trained frame encoder, a dimensionality reduction module and a pooling module. The pooling module uses sliding window approach to slide over the frames of a soccer video clip and then creates actionness scores for each frame. Further, the task of action spotting is performed using non maximum suppression as depicted in Figure 3.11.

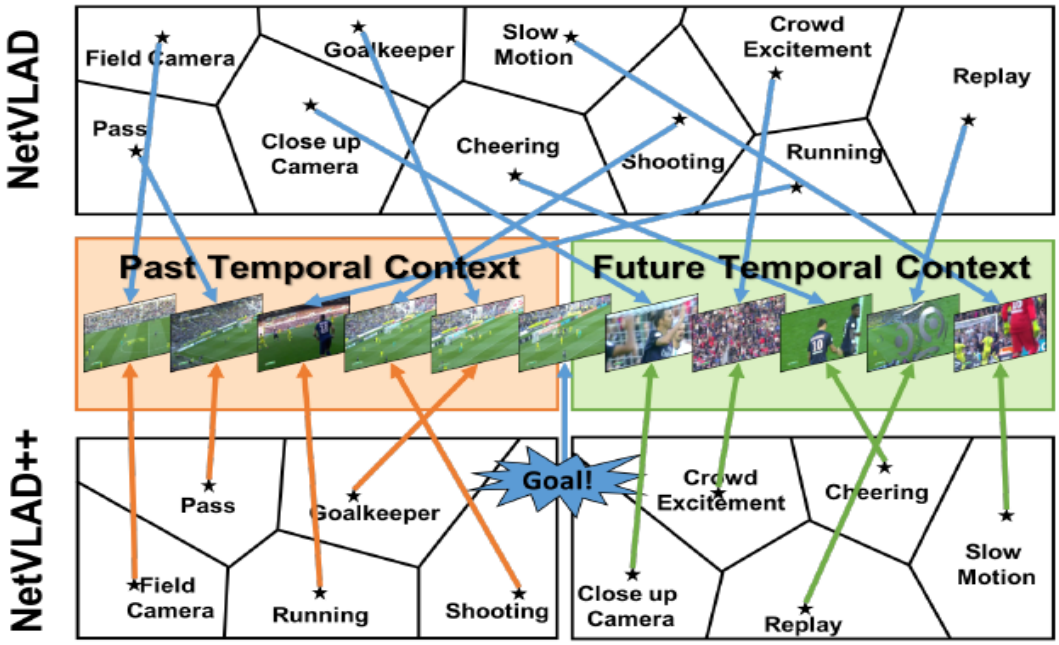

Figure 3.10: NetVLAD and NetVLAD++ pooling modules for action spotting. reprinted from Giancola et al.[31]

Video encoding module

The network uses features extracted from SoccerNet-v2 [17] dataset using ResNet-152 [35] pre-trained on the ImageNet dataset [18]. These features are further used as input for the remaining of the architecture.

Dimensionality reduction module

The dimension size of the SoccerNet-v2 [17] features is reduced from 2048 to 512 using a linear layer as illustrated in the Figure 3.11. The authors argued that having a linear layer to reduce the size of features instead of PCA boosted the performance of architecture [31].

Temporally aware pooling

Input video was divided into multiple window chunks each of T seconds. The features are disentangled into before and after the center of the window frame. These

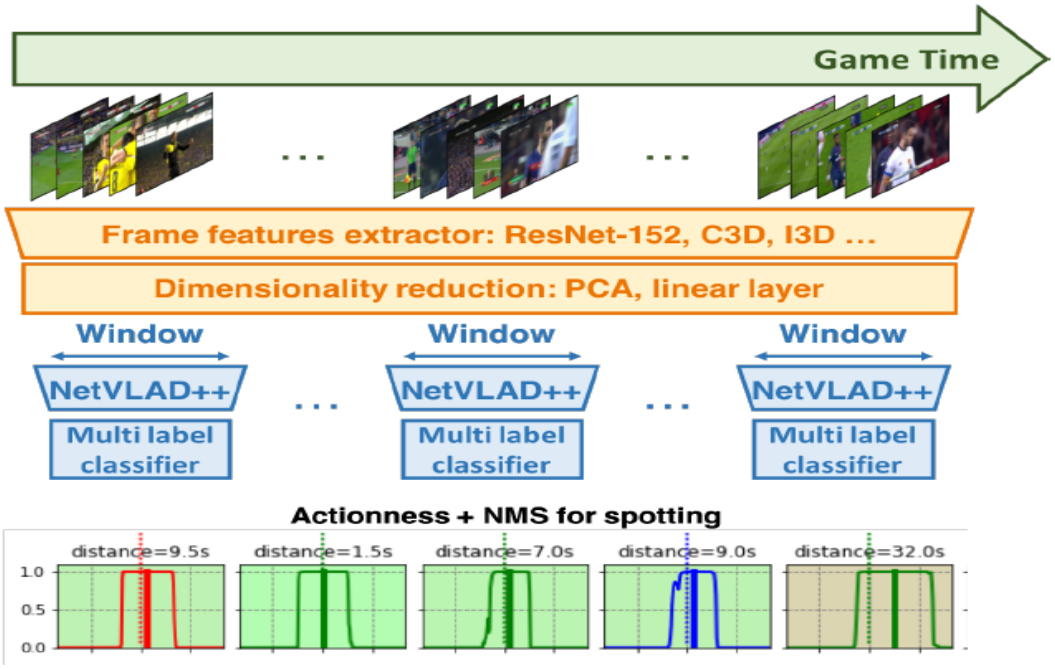

Figure 3.11: Action spotting architecture based on NetVLAD++ pooling module. reprinted from Giancola et al.[31]

set of features are then pooled by the two NetVLAD [31] modules used for each subsets.

Video classification

The network considers non-overlapping video chunks for classifying a video clip using a sliding window of stride T seconds.

3.2.3 Implementation

The experiments were performed with ResNet-152 [36] features and concatenated input features that combine the features concatenated from multiple 3D convolution models. The concatenated features were extracted by using an approach stated by Zhou et al. [74]. The network was fine-tuned by selecting an optimal value of vocabulary size and number of clusters in the NetVLAD++ [31] pooling module. In order to perform our experiments for different configurations, an Nvidia DGX-2

was used which comprises of 16 Nvidia Tesla V100 GPUs each having a RAM of 32 Gigabytes which totals to a capacity of 512 Gigabytes.

The experiment was started with a learning rate of 10^-3 for all variations of the temporally aware pooling method that was decayed from a factor of 10 after the validation did not improve for 10 consecutive epochs. The training is stopped once the learning rate goes beyond 10^-8. An Adam optimizer was used [31] with default β parameters obtained from PyTorch.

3.3 Concatenated fine-tuned input features

During the training of a neural network, it performs the task of feature extraction and computes the values of weights and biases to be passed to the next layer. In case of a classification algorithm, the specified input data is initially passed to a feature extractor module and then the computed outputted features are passed to a classifier network which predicts the final class of the inputted data. When the input data is large and is available in raw format, it is not feasible to train a neural network from scratch as it consumes exceedingly large amount of processing power. In such cases, pre-computed features were used. Pre-computed features are extracted by removing the last classification layer of the network and by using the intermediary features passed to the classification layer as an input for training some other neural network. These features were extracted by using 5 different action recognition models TPN [73], GTA [34], VTN [44], irCSN [63] and I3D-slow [28]. Semantic features of videos were extracted from SoccerNet-v2 [17] dataset by fine-tuning each model as feature-extractor. These feature-extractor models have already been trained for the task of action recognition on large-scale video datasets. The argued stated that the proposed feature combination would significantly improve the performance for the task of action spotting. To perform the fine-tuning of the 5 action recognition models, Zhou et al. [74] performed the training of the video snippets on the fine-tuned action recognition models. Furthermore, the features were concatenated along the feature dimension and are referred to as pre-computed concatenated features. Zhou et al. [74] proposed a two-stage paradigm for the task of action spotting as illustrated

in Figure 3.12. The performance of pre-computed concatenated features to pre-computed single feature extractor approaches was compared and it was observed that using pre computed concatenated features [74] slightly improves the performance of the model along with the fine-tuning of the network.

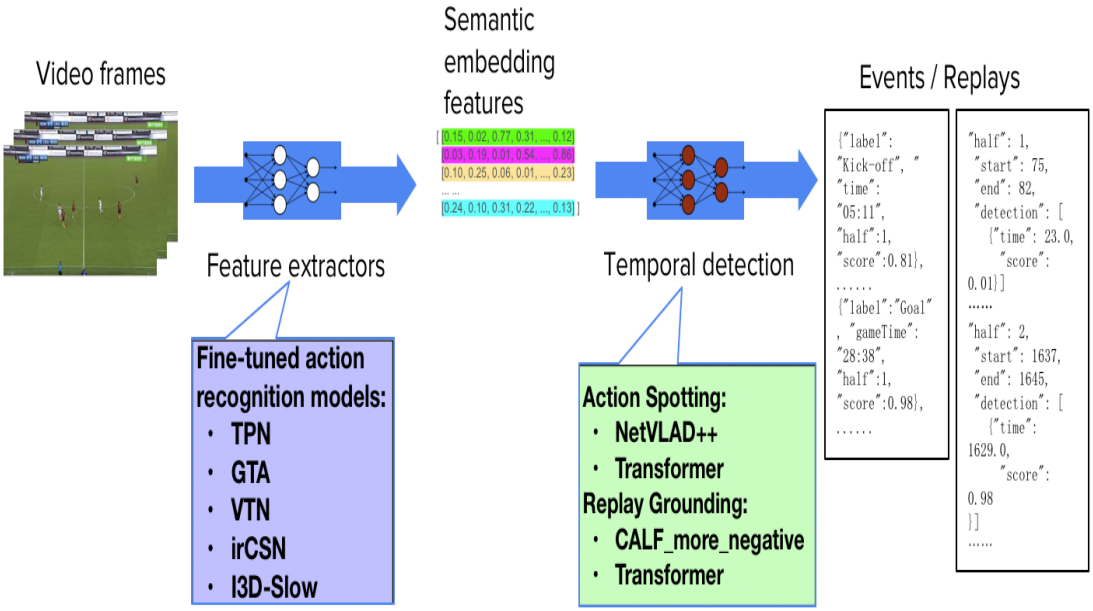

Figure 3.12: Two-stage paradigm for the task of action spotting. Reprinted from [74]

The temporal pyramid network (TPN) [73] can assimilate visual information of different speeds at feature level, and can be efficiently integrated with 2D or 3D backbone networks. The network was trained on Kinetics-400 dataset [41] with a ResNet-101 as backbone and achieved top-1 accuracy of 78.9%. The global temporal attention (GTA) mechanism proposed in [34] incorporates global spatial and temporal interactions in a disjoint fashion [74]. GTA achieves top-1 accuracy of 79.8% on Kinetics-400 dataset [41], with a SlowFast-R101 backbone. Transformer based network forms the basis for the video transformer network (VTN) [44]. VTN comparatively trains faster than 3D CNN networks and when trained on Kinetics-400 dataset [41] can achieve top-1 accuracy of 78.6%. Tran et al. proposed proposed a new network named interaction-reduced channel-separated convolutional networks (irCSN) [63] which segments 3D convolutions further into channel interactions and

spatio-temporal interactions. The network was trained on Kinetics-400 dataset [41] and achieved top-1 accuracy of 82.6%. The I3D-Slow network operates at lower frame rate and captures spatial semantics using a SlowFast framework [28]. The network is further pre-trained with OmniSource [5] dataset and can achieve top-1 accuracy of 80:4% when trained on Kinetics-400 [41]. Below Table 3.2 summarizes the video action recognition models used for extracting the semantic features.

Architecture	**Backbone**	**Dimension**	**Pre-trained on**
TPN [73]	ResNet50/101	2048	Kinetics-400
GTA [34]	ResNet50	2048	Kinetics-400
VTN [44]	ViT-Base	384	Kinetics-400
irCSN [63]	ResNet152	2048	IG65M + Kinetics-400
I3D-Slow [28]	ResNet101	2048	OmniSource

Table 3.2: Video action recognition models for extracting semantic features

3.4 Fine-tuning of hyperparameters

Choosing optimal hyperparameters is a crucial step for a machine learning/ deep learning algorithm. Without the correct set of hyperparameters, the model could either overfit or underfit to the provided training data. In this section, the selected hyperparameters chosen for fine-tuning the temporally aware pooling method are discussed.

3.4.1 CALF

The Context aware loss function proposed by Cioppa et al. [15] was tested on features extracted from ResNet [35] alone and then further on concatenated pre-computed input features as proposed by Zhou et al. [74]. The performance of the model was tested for the task of action spotting. The CALF method was tested with different configurations of chunk size and receptive field of input. The chunk size is defined as the size of the number of consecutive video frames that are fed to the network in seconds. The receptive field is defined as a temporal receptive (in seconds) which

affects the results for a frame more than any other unit of the network. The results with ResNet features and concatenated input features for the context aware loss function method are presented in 4.

3.4.2 Temporally aware pooling method

The temporally aware pooling method also known as NetVLAD ++ proposed by Giancola et al.[31] was tested on features extracted from ResNet [35] and then on concatenated pre-computed input features as proposed by Zhou et al. [74]. The performance of the temporally aware pooling method was tested for the task of action spotting. It was believed that fine-tuning the parameters vocabulary size, number of clusters in the NetVLAD++ pooling module would improve the performance for the task of action spotting. NetVLAD++ pooling module takes N number of features as input each of dimension D and outputs K clusters each of dimension D. Each cluster is then assigned a vocabulary. The NetVLAD++ pooling module disentangles the context in the video into two parts before and after action occurs as illustrated in Figure 3.10. It was believed that increasing the number of clusters in the pooling method would result in assigning more vocabulary to the clusters which in turn would help the pooling method model the context better. The results with ResNet features and concatenated input features for the temporally aware pooling method are presented in the Table.

3.5 Evaluation metrics

A deep learning model can be evaluated against multiple evaluation metrics. The task typically involves training a deep learning model with some input data, then using the trained model to make predictions on a validation dataset not seen by the model before and comparing the predictions made by the model with the ground truth labels. In case of a classification algorithm, evaluation metrics involve comparing the label of the expected class to the predicted class label. Sometimes, a single metric could be misleading and will often cover only a fraction of what is to be understood about the performance of the model. Precision and recall are two very important per-

formance metrics for computing the performance of a model. Precision calculates the number times an accurate prediction of a class occurs per a false prediction of that class [30]. Recall is nothing but the percentage of the data belonging to a particular class which the model correctly predicts as belonging to that class [30]. To calculate the performances of the tested models, standard metrics like accuracy, precision and recall have been used. A *True Positive* is a prediction when the model predicts the correct class, a *False Positive*(FP) when the model predicts an incorrect class, a *True Negative* (TN) when the model correctly rejects a class, and a *False negative*(FN) when the model incorrectly rejects a class. This calculation is defined in the Table 3.3 and is also referred to as a Confusion Matrix.

	Actually Positive	Actually Negative
Predicted Positive	True Positive (TP)	False Positive (FP)
Predicted Negative	False negative (FN)	True Negative (TN)

Table 3.3: Confusion Matrix

Accuracy

Accuracy is defined as the number of correct predictions divided by the total number of predictions.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision

Precision calculates the number times an accurate prediction of a class occurs per a false prediction of that class [30].

$$Precision = \frac{TP}{TP + FP}$$

Recall

Recall is the percentage of the data belonging to a particular class which the model

correctly predicts as belonging to that class [30].

$$Recall = \frac{TP}{TP + FN}$$

For each class, it is considered as a a one-vs-all binary classification problem. The prediction is considered as a true positive only if it falls within a certain tolerance of δ of the ground truth label with a confidence equal to or higher than our set threshold value.

$$|g^{t}{}_{time} - p_{time}| < \frac{\delta}{2}$$

where,
$g^{t}{}_{time}$ is the ground truth spot in seconds
p_{time} is the predicted spot in seconds

The equation specified is used above to create pairs of ground truth spots and predicted spots. Predictions that do not fall under the tolerance of δ are considered as False Positive. If no prediction is made for the given ground truth spot, the it is considered as a False Negative. The calculation of average precision is illustrated in the Figure 3.13.

For example, the two ground truth events are shown in 3.13 and a certain interval is defined in the range of δ. If the prediction falls within the defined interval δ around the ground truth then, it is considered as a True Positive. The range for the interval δ is defined from 5-60 seconds. For example, when δ is 5 seconds, no prediction falls within the interval of 5 seconds. This implies that there are 2 False Positives as there is no prediction that falls within the time interval denoted by arrows and there are 2 False Negatives as there is no prediction that falls inside the time interval. When δ is 10 seconds, there is 1 prediction that falls within the range of the ground truth. However, the other prediction does not fall within the time interval which implies that there is 1 False positive and one False Negative. Similarly, when δ is 15 seconds, both the predictions fall within the time interval which implies that there are 2 True Positives. In this case, the count of the values in the confusion matrix will change de-

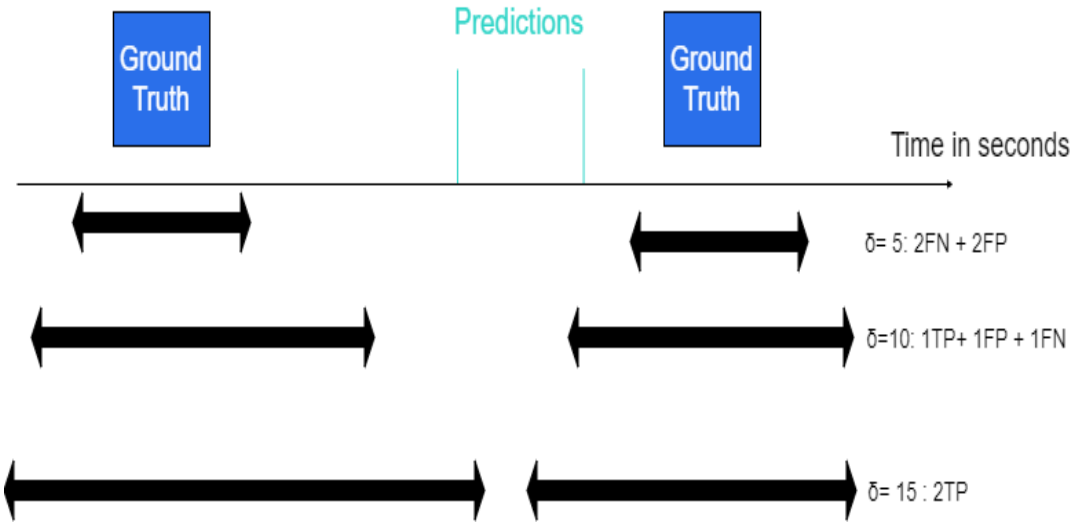

Figure 3.13: Average mAP calculation

pending on the value of δ. For each class based on some threshold confidence score, the values of the confusion matrix are calculated. The predictions that are above the threshold are filtered out. The values of precision and recall per class for the respective δ and threshold confidence score are calculated. Later, the area under the curve on the precision-recall plot is computed which gives the average precision for each class. The value of average precision is calculated for each δ value. The average precision value for each class is calculated and then mean value is calculated of all the average precision values termed as average mAP.

3.6 Summary

In this chapter, the context aware deep learning approaches were discussed and compared using pre-computed single model input features and pre-computed concatenated input features for the task of action spotting on the SoccerNet-v2 dataset. Firstly, the SoccerNet-v2 dataset was introduced and it was further discussed how the pre-computed features were obtained for the dataset. Further, the selected deep learning approaches were discussed in the context of action spotting. The first model proposed a novel Context Aware Loss Function for the task of action spotting. This model was tested using pre-computed single model input features

and pre-computed concatenated input features. The second model proposed a novel pooling method called NetVLAD++ which takes chunks of videos as input and splits the context into before and after an event occurs and then creates clusters out of feature space using k-means clustering and assigns a vocabulary to each cluster. This model was further tested using pre-computed single model input features and pre-computed concatenated input features. Finally, the evaluation metrics like precision, recall and average mAP were presented to evaluate the performance of the selected deep learning approaches.

Chapter 4

Experiments and Results

In Chapter 3, the deep learning approaches were tested using pre-computed features extracted from single model backbones and by concatenating the pre-computed features extracted by combining the features of 5 different models concatenated along the feature dimension. The models were tested using the pre-computed features for the task of action spotting using the SoccerNet-v2 dataset. Multiple experiments were performed on the CALF model with different values of chunk size and tolerance δ. The temporally aware pooling method was tested using single model pre-computed features and by concatenating the pre-computed features extracted by combining the features of 5 different models. further experiments were performed on this model using different values of window size, number of clusters in the pooling method and vocabulary size.

The preliminary experiments performed on the models were presented followed by the experiments performed by fine-tuning with the respective hyperparameters of the models. For the preliminary experiments, the validation set was used to obtain the presented results. Some additional hyperparameters and factors which could affect the results were discussed in this section 4.

4.1 CALF method

Some preliminary experiments were performed to determine the best hyperparameter setting for the CALF method. The results of the preliminary experiments were

discussed in the section 4.

4.1.1 Chunk size and receptive field for CALF

The CALF model was trained with pre-computed ResNet-152 features and pre-computed concatenated features. Results for different hyperparameter setting for the parameters chunk size and receptive field were discussed in this section. The model was tested on the validation set and results were extracted for chunk sizes of 60,120 seconds and receptive field of 40,20 seconds respectively.

Features	Chunk Size	Receptive Field	Average mAP
ResNet-152 Features	120	40	**0.4160**
ResNet-152 Features	60	20	0.3933

Table 4.1: Results on validation set for 2 different configurations of the CALF model trained with single ResNet-152 model as feature extractor backbone. The best score for on the average mAP evaluation metric is highlighted in bold.

The baseline implementation of the CALF method was tested with two different configurations. It can be observed from Table 4.1 that for average mAP, the model with Chunk size of 120 seconds and Receptive field of 40 seconds yield the best results of 41.60 % for the average mAP metric. It was observed that the configuration with Chunk Size of 60 and Receptive field of 20 produced a average mAP score of 39.33 %.

4.1.2 Concatenated features on CALF Method

The baseline implementation of the CALF method was tested with two different configurations with the concatenated pre-computed features extracted from 5 different models as shown in Table 3.2.

It cane be observed from Table 4.2 that for average mAP, the model with Chunk size 120 and Receptive field of 40 yield the best results of 41.83% for the average mAP metric. It was observed that the configuration with Chunk Size of 60 and Receptive

Features	Chunk Size	Receptive Field	Average mAP
Concatenated Features	120	40	**0.4183**
Concatenated Features	60	20	0.3976

Table 4.2: Results on validation set for 2 different configurations of the CALF model trained with concatenated pre-computed features extracted from models shown in Table 3.2. The best score for on the average mAP evaluation metric is highlighted in bold.

field of 20 produced a average mAP score of 39.76 %. It can be observed from the results presented above that there was no significant increase in the score of average mAP metric when the network was trained with pre-computed concatenated features.

4.2 Temporally Aware pooling method

4.2.1 Window and vocabulary size for temporally aware pooling method

The experiments for the temporally aware pooling method were performed with ResNet-152 and pre-computed features. Results with different hyperparameter setting for the parameters window sizes and vocabulary sizes are discussed in this section.The models were tested with a constant window size of 15 and vocabulary sizes of 16, 32, 64 respectively.

Features	Window Size	Vocabulary Size	Average mAP
ResNet-152 Features	15	16	0.4738
ResNet-152 Features	15	32	0.4916
ResNet-152 Features	15	64	**0.5338**

Table 4.3: Results on validation set for 3 different configurations of the temporally aware pooling method trained with single ResNet-152 model as feature extractor backbone. The best score for on the average mAP evaluation metric is highlighted in bold.

The temporally aware pooling method when trained with ResNet-152 features

yielded the best results of 53.38% for average mAP when the window size is 15 and vocabulary size is 64. It was observed that the performance of the model increases with the increase in vocabulary size. The method achieves average mAP of 47.38% and 49.16% when the vocabulary size is 16 and 32 respectively.

4.2.2 Concatenated features on Temporally Aware Pooling method

The experiments for the temporally aware pooling method were performed with pre-computed concatenated features extracted from multiple action recognition models as feature extractor. The models were tested with a constant window size of 15 and vocabulary sizes of 16, 32, 64 respectively.

Features	Window Size	Vocabulary Size	Average mAP
Concatenated Features	15	16	0.4668
Concatenated Features	15	32	0.5149
Concatenated Features	15	64	**0.5587**

Table 4.4: Results on validation set for 3 different configurations of the temporally aware pooling method trained with concatenated pre-computed features extracted from models shown in Table 3.2. The best score for on the average mAP evaluation metric is highlighted in bold.

The temporally aware pooling method when trained with pre-computed concatenated features yielded the best results of 55.87% for average mAP metric for window size 15 and vocabulary size 64. It was observed that the performance of the model increases with the increase in vocabulary size. The method achieves average mAP of 46.68% and 51.49% when the vocabulary size is 16 and 32 respectively.

4.2.3 Class-wise score of average mAP for CALF method

In this section, class-wise results for the CALF method are discussed.

	ResNet-152 Features		Concatenated Features	
	Chunk Size= 120 and Receptive Field = 40	Chunk Size= 60 and Receptive Field = 20	Chunk Size= 120 and Receptive Field = 40	Chunk Size= 60 and Receptive Field = 20
Penalty	**0.3960**	0.3852	0.3982	0.3821
Kick-off	0.3607	0.3688	**0.3711**	0.3539
Goal	**0.7015**	0.6384	0.6989	0.6898
Substitution	0.4985	**0.5432**	0.4722	0.4839
Offside	0.2915	**0.3169**	0.3021	0.2934
Shots on target	**0.2377**	0.2269	0.2236	0.2269
Shots off target	**0.2670**	0.2587	0.2656	0.2434
Clearance	**0.5190**	0.4958	0.5088	0.5069
Ball out of play	0.6640	0.5678	**0.6736**	0.6452
Throw-in	0.5975	0.5587	0.5139	**0.6054**
Foul	**0.5463**	0.5143	0.5425	0.5149
Indirect free-kick	0.4091	**0.4136**	0.4014	0.3917
Direct free-kick	0.4403	0.4321	**0.4406**	0.3961
Corner	**0.7260**	0.7110	0.7019	0.6787
Yellow card	**0.4095**	0.3698	0.3854	0.3566
Red card	0.0077	0.0064	**0.0096**	0.0081
Yellow->red card	0.0	0.0001	**0.0019**	0.0

Table 4.5: Class-wise average mAP scores for CALF method.

4.2.4 Class-wise results of average mAP for temporally aware pooling method

	ResNet-152 Features			Concatenated Features		
	Window = 15 and Vocabulary Size = 16	Window = 15 and Vocabulary Size = 32	Window = 15 and Vocabulary Size = 64	Window = 15 and Vocabulary Size = 16	Window = 15 and Vocabulary Size = 32	Window = 15 and Vocabulary Size = 64
Penalty	0.7312	0.7518	0.7945	0.7215	0.7526	**0.8141**
Kick-off	0.6115	0.6192	0.6132	0.6158	0.6512	**0.6736**
Goal	0.6912	0.7069	0.7318	0.7149	0.7236	**0.7528**
Substitution	0.6512	0.6452	**0.6907**	0.6473	0.6599	0.6669
Offside	0.3691	0.3626	0.3808	0.3752	0.3769	**0.4264**
Shots on target	0.3525	0.3641	0.3895	0.3632	0.3845	**0.3912**
Shots off target	0.3971	0.3941	0.4093	0.3987	**0.4150**	0.4136
Clearance	0.5692	0.5639	**0.5774**	0.5631	0.5741	0.5636
ball out of play	0.6856	0.6912	0.7048	0.6852	0.7136	**0.7159**
Throw-in	0.6628	0.6841	0.6880	0.6741	0.6698	**0.6963**
Foul	0.5983	0.5868	**0.6363**	0.5839	0.6041	0.6123
Indirect free-kick	0.4123	0.3952	0.4443	0.4269	0.4123	**0.4681**
Direct free-kick	0.5743	0.5816	0.5882	0.5921	0.5836	**0.6012**
Corner	0.7739	0.8012	0.8034	0.7843	0.8036	**0.8144**
Yellow card	0.5046	0.5163	0.5685	0.5382	0.5521	**0.5569**
Red card	0.0232	0.0319	0.0374	0.0016	0.0313	**0.0383**
Yellow->red card	0.0123	0.0145	**0.0528**	0.0496	0.0419	0.0521

Table 4.6: Class-wise average mAP scores for Temporally Aware Pooling method. The best score for on the average mAP evaluation metric is highlighted in bold.

4.3 Discussion

In the previous sections, results from our experiments for action spotting with pre-computed ResNet-152 features and pre-computed concatenated features extracted from multiple action recognition approaches were presented.

CALF

It was observed that the CALF method achieved the best results when trained on pre-computed concatenated features and when the chunk size is 120 and receptive field is 40. It was observed that there was only an improvement of 0.15% on the average mAP score on fine-tuning and training with pre-computed concatenated features. The class-wise results were illustrated in Table 4.5. The best average mAP score for each class is highlighted in bold in Table 4.5.

Temporally Aware Pooling

It was observed that the Temporally Aware Pooling method achieved the best results when trained on pre-computed concatenated features and when the window size is 15 and Vocabulary size is 64. It was observed that there was only an improvement of nearly 2.5% on the average mAP score on fine-tuning and training with pre-computed concatenated features. It was observed the average mAP score significantly increased when trained with pre-computed concatenated features. It was further observed that it also benefited when the vocabulary size is increased to 64. The class-wise results were illustrated in Tabletable:TAP class-wise.

4.4 Summary

In this chapter, our results for the task of action spotting on the SoccerNet-v2 dataset [17] were presented. First, the results of the preliminary experiments that were done during the hyperparameter selection in Section 3.4 were presented.Further, the results of using ResNet-152 [35] and pre-computed concatenated features [74] on the CALF [15] and temporally aware pooling method [31] were presented. The best performing

configurations for both the models were discussed and the class-wise results were presented.

Chapter 5

Conclusion

Today, the sports events are manually annotated by human operators. It is a tedious process to annotate these events manually. Automating the process of annotation would have a large impact on the games played in the lower division of the football hierarchy where only limited funds are available. Multiple different type of deep learning algorithms have been used to solve the problem of automatic event detection. Majority of the deep learning approaches proposed in the last decade only use features pre-computed features from a single deep learning model as feature extractor backbone. In this thesis, the deep learning approaches were tested by fine-tuning some of the hyperparameters of the model. The models were further with pre-computed features from 5 different deep learning models as feature extractor backbones.

Our results indicated that deep learning models benefited when tested on concatenated pre-computed features from multiple deep learning models as feature extractor backbones. For the best performing spotting model, the average mAP was nearly 2.5% better with concatenated pre-computed features than with single model feature extractor backbones. There was a significant increase in the average precision for some classes when the pre-computed concatenated features were used over the features from a single model. In total, it was observed that the models were benefited by using concatenated features over features extracted from single model like ResNet-152.

5.1 Main Contributions

The performance of selected aware deep learning approaches was tested for the task of action spotting, as described in 1.2. Following contributions were made to the thesis:

- Multiple different machine learning approaches were researched and tested for the task action spotting. These experiments were conducted with SoccerNet-v2[17] dataset which includes 17 different classes with 300k temporal annotations over soccer broadcast videos. The selected deep learning models were further trained with pre-computed features concatenated from 5 different models.
- The experiments were performed with 2 different deep learning models on SoccerNet-v2 and the performance was analyzed on the test split of the dataset. Multiple configurations of the selected deep learning approaches were tested with various hyperparameters.
- The performances of the selected deep learning approaches were compared using pre-computed concatenated features to pre-computed single feature extractor approaches.There was no significant increase in the performance of the CALF method. However, it was observed that using pre-computed concatenated features improved the performance of the temporally aware pooling model with approximately 2.5% in the average mAP metric.

The results presented in the thesis provide provide valuable to gain insight into using pre-computed concatenated features for training the networks. Additional experiments were performed by fine-tuning some of the hyperparameters which help the aware deep learning approaches model the context better.

5.2 Future Work

To continue the work with pre-computed concatenated features extracted from multiple action recognition models as feature extractor backbones, there are other deep

learning approaches which do not rely on the concept of semantic modelling. It would be beneficial to observe how well the models generalize to using different sports datasets. Further, it would be beneficial to apply this approach of concatenating the input features extracted from multiple models to other sports as well.

Bibliography

[1] May 2021. URL: https://www.youtube.com/watch?v=FksyHTBSI8c&list=LL&index=6.

[2] URL: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[3] Sami Abu-El-Haija et al. *YouTube-8M: A Large-Scale Video Classification Benchmark*. 2016. arXiv: 1609.08675 [cs.CV].

[4] Thomas Alsop. *How many people have access to a computer 2019*. Feb. 2021. URL: https://www.statista.com/statistics/748551/worldwide-households-with-computer/.

[5] Humam Alwassel, Fabian Caba Heilbron and Bernard Ghanem. 'Action Search: Spotting Actions in Videos and Its Application to Temporal Action Localization'. In: *Computer Vision – ECCV 2018*. Ed. by Vittorio Ferrari et al. Cham: Springer International Publishing, 2018, pp. 253–269. ISBN: 978-3-030-01240-3.

[6] Relja Arandjelovic and Andrew Zisserman. 'All about vlad'. In: *2013 IEEE Conference on Computer Vision and Pattern Recognition* (2013). DOI: 10.1109/cvpr.2013.207.

[7] Biagio Brattoli et al. 'Rethinking Zero-Shot Video Classification: End-to-End Training for Realistic Applications'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[8] Matt Brems. *A One-Stop Shop for Principal Component Analysis*. June 2019. URL: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c.

[9] Jason Brownlee. *A gentle introduction to pooling layers for convolutional neural networks*. July 2019. URL: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[10] Jason Brownlee. *Multi-Label Classification with Deep Learning*. Aug. 2020. URL: https://machinelearningmastery.com/multi-label-classification-with-deep-learning/.

[11] Zhe Cao et al. 'Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[12] Joao Carreira and Andrew Zisserman. 'Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[13] Joao Carreira et al. *A Short Note about Kinetics-600*. 2018. arXiv: 1808.01340 [cs.CV].

[14] Joao Carreira et al. *A Short Note on the Kinetics-700 Human Action Dataset*. 2019. arXiv: 1907.06987 [cs.CV].

[15] Anthony Cioppa et al. *A Context-Aware Loss Function for Action Spotting in Soccer Videos*. 2020. arXiv: 1912.01326 [cs.CV].

[16] Navneet Dalal, Bill Triggs and Cordelia Schmid. 'Human Detection Using Oriented Histograms of Flow and Appearance'. In: *Computer Vision – ECCV 2006*. Ed. by Aleš Leonardis, Horst Bischof and Axel Pinz. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 428–441. ISBN: 978-3-540-33835-2.

[17] Adrien Deliège et al. *SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos*. 2021. arXiv: 2011.13367 [cs.CV].

[18] Jia Deng et al. 'ImageNet: A large-scale hierarchical image database'. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[19] P.J. Denning et al. 'Computing as a discipline'. In: *Computer* 22.2 (1989), pp. 63–70. DOI: 10.1109/2.19833.

[20] By: IBM Cloud Education. *What are Convolutional Neural Networks?* URL: https://www.ibm.com/cloud/learn/convolutional-neural-networks.

[21] By: IBM Cloud Education. *What are Neural Networks?* URL: https://www.ibm.com/cloud/learn/neural-networks.

[22] By: IBM Cloud Education. *What is Overfitting?* URL: https://www.ibm.com/cloud/learn/overfitting.

[23] A. Ekin, A.M. Tekalp and R. Mehrotra. 'Automatic soccer video analysis and summarization'. In: *IEEE Transactions on Image Processing* 12.7 (2003), pp. 796–807. DOI: 10.1109/TIP.2003.812758.

[24] Bernard Ghanem Fabian Caba Heilbron Victor Escorcia and Juan Carlos Niebles. 'ActivityNet: A Large-Scale Video Benchmark for Human Activity Understanding'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 961–970.

[25] Christoph Feichtenhofer. 'X3D: Expanding Architectures for Efficient Video Recognition'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[26] Christoph Feichtenhofer, Axel Pinz and Richard P. Wildes. *Spatiotemporal Residual Networks for Video Action Recognition*. 2016. arXiv: 1611.02155 [cs.CV].

[27] Christoph Feichtenhofer, Axel Pinz and Andrew Zisserman. 'Convolutional Two-Stream Network Fusion for Video Action Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[28] Christoph Feichtenhofer et al. 'SlowFast Networks for Video Recognition'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[29] *Forzify*. URL: https://www.forzasys.com/Forzify.html.

[30] Datascience George. *The precision-recall trade-off*. June 2020. URL: https://datascience-george.medium.com/the-precision-recall-trade-off-aa295faba140.

[31] Silvio Giancola and Bernard Ghanem. *Temporally-Aware Feature Pooling for Action Spotting in Soccer Broadcasts*. 2021. arXiv: 2104.06779 [cs.CV].

[32] Silvio Giancola et al. 'SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos'. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (June 2018). DOI: 10.1109/cvprw.2018.00223. URL: http://dx.doi.org/10.1109/CVPRW.2018.00223.

[33] MA Goodale and AD Milner. 'Separate visual pathways for perception and action'. In: *Trends in neurosciences* 15.1 (Jan. 1992), pp. 20–25. ISSN: 0166-2236. DOI: 10.1016/0166-2236(92)90344-8. URL: https://doi.org/10.1016/0166-2236(92)90344-8.

[34] Bo He et al. *GTA: Global Temporal Attention for Video Action Understanding*. 2021. arXiv: 2012.08510 [cs.CV].

[35] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[36] Kaiming He et al. 'Deep Residual Learning for Image Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[37] Samuel Hurault, Coloma Ballester and Gloria Haro. 'Self-Supervised Small Soccer Player Detection and Tracking'. In: *Proceedings of the 3rd International Workshop on Multimedia Content Analysis in Sports* (Oct. 2020). DOI: 10.1145/3422844.3423054. URL: http://dx.doi.org/10.1145/3422844.3423054.

[38] Haroon Idrees et al. 'The THUMOS challenge on action recognition for videos "in the wild"'. In: *Computer Vision and Image Understanding* 155 (Feb. 2017), pp. 1–23. ISSN: 1077-3142. DOI: 10.1016/j.cviu.2016.10.018. URL: http://dx.doi.org/10.1016/j.cviu.2016.10.018.

[39] Herve Jegou et al. 'Aggregating local descriptors into a compact image representation'. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2010). DOI: 10.1109/cvpr.2010.5540039.

[40] Andrej Karpathy et al. 'Large-scale Video Classification with Convolutional Neural Networks'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[41] Will Kay et al. *The Kinetics Human Action Video Dataset*. 2017. arXiv: 1705.06950 [cs.CV].

[42] H. Kuehne et al. 'HMDB: A large video database for human motion recognition'. In: *2011 International Conference on Computer Vision*. 2011, pp. 2556–2563. DOI: 10.1109/ICCV.2011.6126543.

[43] Behzad Mahaseni, Erma Rahayu Mohd Faizal and Ram Gopal Raj. 'Spotting Football Events Using Two-Stream Convolutional Neural Network and Dilated Recurrent Neural Network'. In: *IEEE Access* 9 (2021), pp. 61929–61942. DOI: 10.1109/ACCESS.2021.3074831.

[44] Daniel Neimark et al. *Video Transformer Network*. 2021. arXiv: 2102.00719 [cs.CV].

[45] Olav A. Norgård Rongved et al. 'Real-Time Detection of Events in Soccer Videos using 3D Convolutional Neural Networks'. In: *2020 IEEE International Symposium on Multimedia (ISM)*. 2020, pp. 135–144. DOI: 10.1109/ISM.2020.00030.

[46] *Overfitting and underfitting*. URL: https://www.educative.io/edpresso/overfitting-and-underfitting.

[47] Florent Perronnin and Christopher Dance. 'Fisher Kernels on Visual Vocabularies for Image Categorization'. In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383266.

[48] Jatin Prakash. *Non maximum suppression: Theory and implementation in pytorch*. June 2021. URL: https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/.

[49] Zhaofan Qiu, Ting Yao and Tao Mei. 'Learning Spatio-Temporal Representation With Pseudo-3D Residual Networks'. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[50] Joseph Redmon et al. 'You Only Look Once: Unified, Real-Time Object Detection'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[51] *Regression Analysis*. July 2021. URL: https://corporatefinanceinstitute.com/resources/knowledge/finance/regression-analysis/.

[52] Published by S. O'Dea and Aug 6. *Smartphone users 2026*. Aug. 2021. URL: https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/.

[53] *Selecting the right bounding box using non-max suppression (with implementation)*. Aug. 2020. URL: https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/.

[54] Zheng Shou, Dongang Wang and Shih-Fu Chang. 'Temporal Action Localization in Untrimmed Videos via Multi-Stage CNNs'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[55] Karen Simonyan and Andrew Zisserman. *Two-Stream Convolutional Networks for Action Recognition in Videos*. 2014. arXiv: 1406.2199 [cs.CV].

[56] *Soccer is a sport in the world*. URL: https://www.bartleby.com/essay/Soccer-Is-A-Sport-In-The-World-FJV5YW2U49V.

[57] Khurram Soomro and Amir R. Zamir. 'Action Recognition in Realistic Sports Videos'. In: *Computer Vision in Sports Advances in Computer Vision and Pattern Recognition* (2014), pp. 181–208. DOI: 10.1007/978-3-319-09396-3_9.

[58] Khurram Soomro, Amir Roshan Zamir and Mubarak Shah. *UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild*. 2012. arXiv: 1212.0402 [cs.CV].

[59] Matteo Tomei et al. *RMS-Net: Regression and Masking for Soccer Event Spotting*. 2021. arXiv: 2102.07624 [cs.CV].

[60] Lisa A. Torrey and J. Shavlik. 'Chapter 11 Transfer Learning'. In: 2009.

[61] Du Tran et al. 'A Closer Look at Spatiotemporal Convolutions for Action Recognition'. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[62] Du Tran et al. 'Learning Spatiotemporal Features With 3D Convolutional Networks'. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[63] Du Tran et al. 'Video Classification With Channel-Separated Convolutional Networks'. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[64] G. Tsagkatakis, M. Jaber and P. Tsakalides. 'Goal!! Event detection in sports video'. In: *Electronic Imaging* (2017). DOI: 10.2352/issn.2470-1173.2017.16.cvas-344.

[65] Bastien Vanderplaetse and Stéphane Dupont. *Improved Soccer Action Spotting using both Audio and Video Streams*. 2020. arXiv: 2011.04258 [cs.CV].

[66] Kanav Vats et al. 'Event Detection in Coarsely Annotated Sports Videos via Parallel Multi-Receptive Field 1D Convolutions'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. June 2020.

[67] *Video Assistant Referee (VAR) Technology*. URL: https://www.fifa.com/technical/football-technology/standards/video-assistant-referee.

[68] Heng Wang and Cordelia Schmid. 'Action Recognition with Improved Trajectories'. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. Dec. 2013.

[69] Heng Wang et al. 'Dense trajectories and motion boundary descriptors for action recognition'. In: *International Journal of Computer Vision* 103.1 (May 2013), pp. 60–79. DOI: 10.1007/s11263-012-0594-8. URL: https://hal.inria.fr/hal-00803241.

[70] Limin Wang et al. 'Actionness estimation using hybrid fully convolutional networks'. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). DOI: 10.1109/cvpr.2016.296.

[71] Limin Wang et al. 'Temporal Segment Networks: Towards Good Practices for Deep Action Recognition'. In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Cham: Springer International Publishing, 2016, pp. 20–36. ISBN: 978-3-319-46484-8.

[72] Chao-Yuan Wu et al. *A Multigrid Method for Efficiently Training Video Models*. 2020. arXiv: 1912.00998 [cs.CV].

[73] Ceyuan Yang et al. 'Temporal Pyramid Network for Action Recognition'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2020.

[74] Xin Zhou et al. *Feature Combination Meets Attention: Baidu Soccer Embeddings and Transformer based Temporal Detection*. 2021. arXiv: 2106.14447 [cs.CV].

[75] Mohammadreza Zolfaghari, Kamaljeet Singh and Thomas Brox. 'ECO: Efficient Convolutional Network for Online Video Understanding'. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Sept. 2018.

Appendix A

Project Code Base

The following link contains implementation of the deep learning models in python language.

- https://drive.google.com/drive/folders/15QJLU3Iq2fnjhAK6cy7xs6psg5RF8hY9?usp=sharing