Lab Report
NetVLAD++ Feature Pooling
for Action Spotting in Soccer Broadcasts

1 Introduction

In this project, I implemented one of the solutions to the task of Action Spotting on SoccerNet benchmark dataset. It was first introduced in [4] and is composed of 500 complete soccer games from six main European leagues, covering three seasons from 2014 to 2017 and a total duration of 764 hours. The annotations for this dataset were later largely extended and presented as SoccerNet-v2 in [2]. This is the version of the dataset used in this project.

This project implements NetVLAD++ feature pooling architecture presented in [3] and according to this page for SoccerNet-v2 dataset on Paperswithcode website is current state of the art model on this dataset.

2 Task description

2.1 Action Spotting definition

In order to understand the actions of a broadcast soccer game, SoccerNet introduces the task of action spotting, which consists in finding all the actions occurring in the videos. In this task, the actions are anchored with a single timestamp, contrary to the task of activity localization, where activities are delimited with start and stop timestamps.

Spotting consists of finding the anchor time (or spot) that identifies an event. Intuitively, the closer the candidate spot is from the target, the better the spotting is, and its capacity is measured by its distance from the target. Since perfectly spotting a target is intrinsically arduous, they introduce a tolerance δ within which a event is considered to be spotted (hit) by a candidate. They believe that event spotting is better defined and easier than detection since it focuses only on identifying the presence of an event within a given tolerance. An iterative process can refine such tolerance at will by using fine localization methods around candidate spots.

2.2 Metric used for evaluation

By introducing the task of spotting, they also define the metric to be used for evaluation.
First of all, we define a candidate spot as positive if it lands within a tolerance δ around the anchor of an event. For each tolerance, we can recast the spotting problem as a general temporal detection problem, where the tIoU threshold used is very small. In that case, we can compute the recall, precision, Average Precision (AP) for each given class, and a mean Average Precision (mAP) across all classes. For general comparison, we also define an Average-mAP over a set of predefined δ tolerances, in this dataset ranging from 5 to 60 seconds.

2.3 Dataset description

SoccerNet-v2 stands out as one of the largest overall, and the largest for soccer videos by far. In particular, authors manually annotated 300k timestamps, temporally anchored in the 764 hours of the 500 games of SoccerNet [4]. Authors identify 17 types of actions from the most important in soccer:
`'Penalty', 'Kick-off', 'Goal', 'Substitution', 'Offside', 'Shots on target',`
`'Shots off target', 'Clearance', 'Ball out of play', 'Throw-in', 'Foul',`
`'Indirect free-kick', 'Direct free-kick', 'Corner',`
`'Yellow card', 'Red card', 'Yellow->red card'.`

They annotate each action of the 500 games of SoccerNet with a single timestamp, defined by well-established soccer rules. For instance, for a corner, the last frame of the shot is annotated, i.e. showing the last contact between the player's foot and the ball.

Additionally, they enrich each timestamp with a binary visibility tag that states whether the associated action is shown in the broadcast video or unshown, in which case the action must be inferred by the viewer. For example, this

happens when the producer shows a replay of a shot off target that lasts past the clearance shot of the goalkeeper: the viewer knows that the clearance has been made despite it was not shown on the TV broadcast. Spotting unshown actions is challenging because it requires a fine understanding of the game, beyond frame-based analysis, as it forces to consider the temporal context around the actions.

3 Model overivew

3.1 Description of NetVLAD++

NetVLAD++ [3] is a temporally-aware modification of NetVLAD [1], which is a differentiable pooling technique inspired by VLAD [5].

VLAD. Formally, given a set of N D-dimensional features $\{x_i\}_{i=1..N}$ as input, a set of K clusters centers $\{c_k\}_{k=1..K}$ with same dimension D as VLAD parameters, the output of the VLAD descriptor V is defined by:

$$V(j,k) = \sum_{i=1}^{N} a_k(\mathbf{x}_i)(\mathbf{x}_i(j) - \mathbf{c}_k(j)) \tag{1}$$

where $\mathbf{x}_i(j)$ and $\mathbf{c}_k(j)$ are respectively the j-th dimensions of the i-th descriptor and k-th cluster center. $a_k(\mathbf{x}_i)$ denotes the hard assignment of the sample $\mathbf{x}_i$ from its closer center, i.e. $a_k(\mathbf{x}_i) = 1$ if $\mathbf{c}_k$ is the closest center of $\mathbf{x}_i$, 0 otherwise. The matrix V is then L2-normalized at the cluster level, flatten into a vector of length $D \times K$ and further L2-normalized globally.

NetVLAD. The VLAD module is non-differentiable due to the hard assignment $a_k(\mathbf{x}_i)$ of the samples $\{\mathbf{x}_i\}_{i=1}^{N}$ to the clusters $\{\mathbf{c}_k\}_{k=1}^{K}$. Those hard-assignment creates discontinuities in the feature space between the clusters, impeding gradients to flow properly. To circumvent this issue, NetVLAD [1] introduces a soft-assignment $\tilde{a}_k(\mathbf{x}_i)$ for the samples $\{\mathbf{x}_i\}_{i=1}^{N}$, based on their distance to each cluster center. Formally:

$$\tilde{a}_k(\mathbf{x}_i) = \frac{e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_k\|^2}}{\sum_{k'=1}^{K} e^{-\alpha\|\mathbf{x}_i - \mathbf{c}_{k'}\|^2}} \tag{2}$$

$\tilde{a}_k(\mathbf{x}_i)$ ranges between 0 and 1, with the highest value assigned to the closest center. α is a temperature parameter that controls the softness of the assignment, a high value for α (e.g. $\alpha \to +\infty$) would lead to a hard assignment like in VLAD. Furthermore, by expanding the squares and noticing that $e^{-\alpha\|\mathbf{x}_i\|^2}$ will cancel between the numerator and the denominator, we can interpret Equation (2) as the softmax of a convolutional layer for the input features parameterized by $\mathbf{w}_k = 2\alpha\mathbf{c}_k$ and $b_k = -\alpha\|\mathbf{c}_k\|^2$. Formally:

$$\tilde{a}_k(\mathbf{x}_i) = \frac{e^{\mathbf{w}_k^T \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}}} \tag{3}$$

Finally, by plugging the soft-assignment from (3) into the VLAD formulation in (1), the NetVLAD features are defined as in Equation (4), later L2-normalized per cluster, flattened and further L2-normalized in its entirety.

$$V(j,k) = \sum_{i=1}^{N} \frac{e^{\mathbf{w}_k^T \mathbf{x}_i + b_k}}{\sum_{k'} e^{\mathbf{w}_{k'}^T \mathbf{x}_i + b_{k'}}} (\mathbf{x}_i(j) - \mathbf{c}_k(j)) \tag{4}$$

Note that the original VLAD optimizes solely the cluster centers $\mathbf{c}_k$, while NetVLAD optimizes for $\mathbf{w}_k$, b_k and $\mathbf{c}_k$ independently, dropping the constraint of $\mathbf{w}_k = 2\alpha\mathbf{c}_k$ and $b_k = -\alpha\|\mathbf{c}_k\|^2$. These constraints were dropped for further freedom in the training process.

NetVLAD++. The VLAD and NetVLAD pooling methods are permutation invariant, as a consequence, do not consider the order of the frames, but only aggregates the features as a set. In the particular case of action spotting, the frames features from the videos are temporally ordered in time, and can be categorized between past and future context.

The amount of context embedded before and after an action occurs is different, yet complementary. In addition, different actions might share similar vocabulary either before or after those actions occur, but usually not both. As an example, the semantic information contained before a "goal" occurs and before a "shot on/off target" occurs are similar, representing a lower level semantic concept of a player shooting on a target and a goalkeeper trying to catch that ball. Yet, those two action classes depict different contextual semantics after it occurs, with the presence of cheering (for "goal") or frustration (for simple "shot") in the players. Following a similar logic, the spotting of a "penalty" would benefit more from the knowledge of what happened before that penalty was shot, as the follow-up cheering would look similar to any other goal. Without loss of generality, it appears that

the amount of information to pool among the features before and after an action occurs might contain different low-level semantics, helping identifying specific fine-grained actions.

To that end, a novel temporally-aware pooling module **NetVLAD++** is used. In particular, it learns 2 different NetVLAD pooling modules for the frame features from before and after an action occurs. Authors define the past context as the frame feature with a temporal offset in $[-T_b, 0)$ and the future context as the frame feature with a temporal offset in $[0, T_a]$. Each pooling module aggregates different clusters of information from the 2 subsets of features, using K_a and K_b clusters, respectively for the after and before subsets. Formally:

$$V = \mathcal{V}(V_b, V_a) \tag{5}$$

with $\mathcal{V}$ an aggregation of V_b and V_a that represent the NetVLAD pooled features for the sample before and after the action occurs, parameterized with K_b clusters for the past context and K_a clusters for the future context.

4 Architecture for Action Spotting

The architecture is based on a pre-trained frame feature encoder, a dimensionality reduction, a pooling module from a temporally sliding window and a per-frame classifier that depicts a class-aware actionness. The action spotting is then performed using a non-maximum suppression (NMS).

4.1 Video encoding

I used the features extracted by SoccerNet-v2 dataset authors, based on ResNet-152 pre-trained on ImageNet. They are availabe for downloading from their Python pachage (https://pypi.org/project/SoccerNet/). The weights are frozen and the frame features are pre-extracted at 2fps with a resolution of 224x224, scaled down in height then cropped on the sides for the width. The features correspond to the activation of the last layer of the ResNet-152 architecture, after the max pooling across the 2D feature map and before the classification layer, resulting in features of dimension 2048.

Important Note: In addition to providing 2048-dimentional frame features, authors also provide 512-dimentional features, which were obtained by applying PCA reduction to the original features. These features are used in their first work [4]. in their NetVLAD++ paper [3] authors show that applying a linear layer to 2048-dimentional features provides huge benefit, because a linear layer would learn a better linear combination of the frame features, by removing the orthogonality constraint introduced by PCA. Regardless of that, I am using 512-dimentional features due to computational constraints (in particular, lack of VRAM in my GPU and space in SSD to store them). For this reason, my results are worse than the best results in [3], although they are the same as the results in that paper, where they describe the difference between PCA and linear layer.

4.2 Temporally-aware pooling

We consider window chunks of time T seconds along the video. The temporally contiguous set of features are split equally before and after the center of the window and pooled accordingly. We normalize the features along the feature dimension and apply the 2 NetVLAD module for each subset of features. The 2 output NetVLAD features are concatenated along the feature dimension, leading into a feature of dimension $(K_b + K_a) \times D$.

4.3 Video Chunk Classification

In training, we consider nonoverlapping window chunks with a sliding window of stride T. We build a classifier on top of the pooled features, composed of a single neural layer with sigmoid activation and dropout. Since multiple actions can occur in the same temporal window, we consider a multi-label classification approach. A video chunk is labeled with all classes that appear on the chunk with a multi-label one-hot encoding. We optimize for a multi-label binary cross-entropy loss as defined:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} y_i \log(x_n) + (1 - y_i) \log(1 - x_n) \tag{6}$$

Inference. We run the sliding window of time T along unseen videos with a temporal stride of 1 to report the classaware actionness scores in time. We use a Non Maximum Suppression (NMS) module to reduce positive spots closer than a given temporal threshold T_{NMS}.

5 Results

5.1 Optimal hyperparameters

All hyperparametes are taken directly from [3]. The paper describes the process of hyperparameter tuning in great details.

Highest performances are reached by setting a temporal window $T = 15$s and $K = 64$ clusters and considering the same amount of temporal context from before and after the actions, so that $T_b = T_a = T/2$ and $K_b = K_B = K/2$. Finally, we suppress duplicate spottings around the highest confidence score with a NMS considering a centered window of $T_{NMS} = 30$s.

We use the Adam optimizer with default β parameters from PyTorch and a starting learning rate of 10^{-3} that we decay from a factor of 10 after the validation loss does not improve for 10 consecutive epochs. We stop the training once the learning rate decays below 10^{-8}.

5.2 Final metrics on test set

The main results are provided in the table below. As noted before, due to lack of computational resources I was not able to reproduce SOTA results from [3], but my results are still competitive, and, in fact, almost the same as results from paper that use PCA-d 512-dimentional features.

	Average	visible	unshown	Ball out	Throw-in	Foul	Ind. free-kick	Clearance	Shots on tar.	Shots off tar.	Corner	Substitution	Kick-off	Yellow card	Offside	Dir. free-kick	Goal	Penalty	Yel.→Red	Red card
Best in [3]	53.3	59.1	35.1	70.2	68.9	64.1	45.2	56.6	38.2	40.4	79.8	68.9	61.1	56.1	38.0	58.2	71.6	79.1	5.5	3.5
PCA in [3]	50.7	55.8	37.3	68.2	65.3	62.4	43.4	56.0	37.1	38.3	78.9	70.3	59.6	50.0	35.3	55.2	70.2	67.7	1.7	1.5
My res.	50.6	55.6	37.1	68.6	65.2	62.2	43.4	55.7	37.3	38.8	78.9	69.9	60.0	49.4	35.3	55.5	70.2	66.6	1.8	1.2

References

[1] Relja Arandjelović et al. *NetVLAD: CNN architecture for weakly supervised place recognition*. 2015. DOI: 10.48550/ARXIV.1511.07247. URL: https://arxiv.org/abs/1511.07247.

[2] Adrien Deliège et al. *SoccerNet-v2: A Dataset and Benchmarks for Holistic Understanding of Broadcast Soccer Videos*. 2020. DOI: 10.48550/ARXIV.2011.13367. URL: https://arxiv.org/abs/2011.13367.

[3] Silvio Giancola and Bernard Ghanem. *Temporally-Aware Feature Pooling for Action Spotting in Soccer Broadcasts*. 2021. DOI: 10.48550/ARXIV.2104.06779. URL: https://arxiv.org/abs/2104.06779.

[4] Silvio Giancola et al. *SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos*. June 2018. DOI: 10.1109/cvprw.2018.00223. URL: https://arxiv.org/abs/1804.04527.

[5] Hervé Jégou et al. *Aggregating local descriptors into a compact image representation*. 2010. DOI: 10.1109/CVPR.2010.5540039.