AN ABSTRACT OF THE THESIS OF

Dimitrios Trigkakis for the degree of Master of Science in Computer Science presented on December 5, 2017.

Title: A Deep Action Segmentation and Its Explanation with a Dictionary of Meaningful Attention Maps

Abstract approved: _____

Sinisa Todorovic

This thesis addresses the problem of temporal action segmentation in videos, where the goal is to label every video frame with the appropriate action class present. We focus on the domain of NFL football videos, where action classes represent common football play types. For action segmentation, we use a temporal convolutional network (TCN) that accounts for temporal context for labeling every frame, given a sequence frame deep features as input. Toward better understanding the TCN's decision-making process, we also compute TCNs visual attention maps at the pixel level with the excitation back-propagation algorithm. As the attention maps highlight the most discriminative frame parts, we hypothesize that frequent space-time patterns of attention maps correspond to meaningful concepts in the football domain (e.g., characteristic spatial formation of the players, part of a yard line). Thus, computing the attention maps and discovering a dictionary of their frequent patterns could be used to provide meaningful explanations about TCNs predictions. For dictionary learning, we first use a deep spatio-temporal auto-encoder to project the input attention maps onto a latent (encoded) feature space, where the attention maps can then be clustered. The resulting clusters can be assigned a semantic meaning by visual inspection for subsequent explanations about TCNs performance. We present a quantitative evaluation of TCN on challenging NFL football videos, and visualizations of the discovered dictionary of discriminative frame parts.

A Deep Action Segmentation and Its Explanation with a Dictionary of Meaningful Attention Maps

by

Dimitrios Trigkakis

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented December 5, 2017
Commencement June 2018

Master of Science thesis of Dimitrios Trigkakis presented on December 5, 2017.

APPROVED:

Major Professor, representing Computer Science

Director of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Dimitrios Trigkakis, Author

ACKNOWLEDGEMENTS

I would like to thank my advisor, Dr. Sinisa Todorovic, for his constant support and guidance. My committee members, Fuxin Li, Mike Rosulek and Alan Fern for their input and help throughout my studies. Fellow students Trevor Fiez, Xu Xu and Sam Greydanus, who have been instrumental to the initial phases of this work. Colleagues Michael Lam and Peng Lei for their help throughout my studies. Finally my family and friends who have helped me with the challenges of graduate life.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

Figure Page

LIST OF FIGURES (Continued)

Figure Page

LIST OF TABLES

Chapter 1: Introduction

In this work our goal is two-fold. Firstly, temporally segment videos into intervals of activity from one of the classes of interest. Secondly, generate visualizations of the most important video parts for the prediction. We focus on the domain of NFL (National Football League) football videos [1], for the task of segmenting videos into three distinct viewpoints and classifying each resulting segment by the playtype. For these tasks, the viewpoint types include 'front view', 'side view' and 'scoreboard'. The playtypes include 'Rush', 'Pass', and 'Other'.

Temporal video segmentation has a wide range of applications, including rapid extraction of video intervals, automatic annotation of videos for browsing and retrieval as well as video decomposition to scenes for further processing. Deep learning approaches have made significant advances in video segmentation [10][9]. However, predictions of deep neural networks are hard to understand by a human. Recent research has attempted to provide explanations of the predictions made by deep models [21][15]. One of the goals in this research is to also enable providing human-readable explanations for the classification decisions of neural networks with minimal human intervention. Our work is an initial step towards this goal.

1.1 Challenges

The domain of NFL football videos presents many challenges. There are large differences among videos due to a multitude of factors, including variations in player appearance, appearance of the field, as well as stadium-specific characteristics such as logos on the field and color of grass. Additional challenges include variations in lighting conditions and camera viewing angles.

The dataset of the NFL videos that we consider has a non-uniform distribution of target classes for both viewpoints and playtypes. This data imbalance makes the training of classifiers difficult. There are also video artifacts like noisy frames and sudden camera movements. For current video analysis systems, variations in camera viewpoints

typically present the greatest challenge. Therefore in this work we make our video analysis viewpoint invariant by first identifying the viewpoint and then applying the suitable classifier for this viewpoint. Thus our video analysis consists of two steps, viewpoint segmentation and playtype classification.

1.2 Contributions

To our knowledge, this is the first work on visualizations aimed at explaining the segmentation of football videos. We also compiled a new dataset of NFL videos and manually annotated video intervals by viewpoints and playtypes.

1.3 Overview of our Approach

Video segmentation in our approach is addressed in two steps. We first detect intervals of different viewpoints and then classify each with a playtype class. This is illustrated in Figure 1.1. The viewpoint segmentation module partitions the input NFL video into viewpoint segments.

We consider three types of viewpoints: a scoreboard view, a side view and a front view of the field. Some examples of these views are shown in Figure 1.2.

For viewpoint segmentation, we use the VGG network [16] for predicting the viewpoint class of each frame. Then we use the Encoder-Decoder Temporal Convolutional Network (ED-TCN) [10] to smooth out the resulting frame predictions into temporally coherent segments. The second step of our approach is the playtype classification module which classifies previously segmented video intervals into three categories of plays, ‘Pass‘, ‘Rush‘ and ‘Other‘ as shown in Figure 1.3. The playtypes in the ‘Other‘ category are underrepresented in the dataset and have been grouped together. The category ‘Other‘ includes playtypes such as ‘Kickoff‘, ‘Punt‘ and ‘Field Goal‘ among others. The playtype classifier uses the VGG network for feature extraction for each frame as well as ED-TCN that fuses VGG features from all frames and predicts the playtype.

Toward explaining the above predictions, we also develop an explaining variational auto-encoder which we call X-VAE. Figure 1.5 shows our X-VAE model. The inputs to this model are attention maps, the most discriminative parts of video frames. An

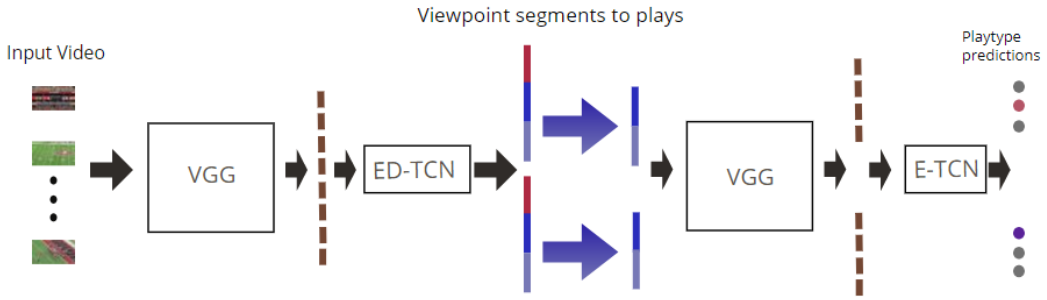

Figure 1.1: Overview of the Video Analysis Architecture. Given a sequence of frames from an input video, we extract deep features from every frame using a convolutional network. We make per-frame predictions for the viewpoint classes by applying the ED-TCN model over the input features. We obtain viewpoint segments for every consecutive sequence of frames with the same class. By separating the sequences between scoreboard views, we obtain new frame sequences, each one corresponding to a single play. We perform the same procedure, using a modified version of ED-TCN that produces a single prediction for every play sequence. The final predictions correspond to the playtype classes under consideration.

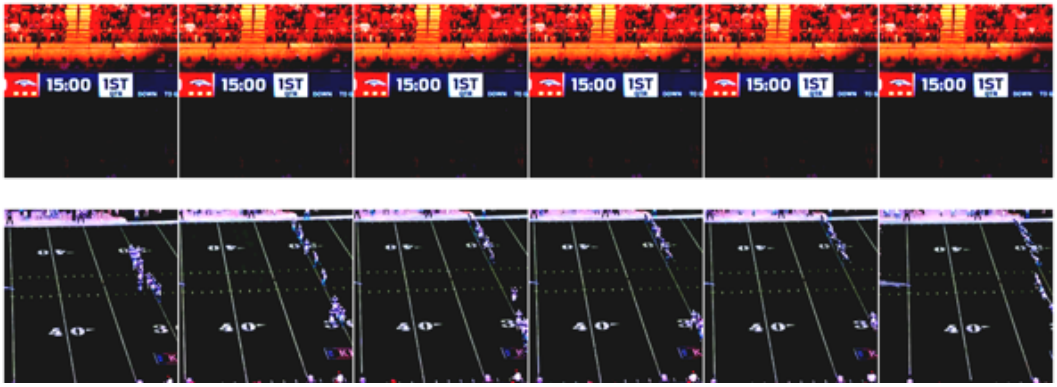

Figure 1.2: Sample input for viewpoint segmentation

attention map includes areas in a frame that give rise to high-activations in the network. Visualizing these attention maps to the user provides an explanation of the prediction in terms of what areas are responsible for the prediction. For example, if the attention

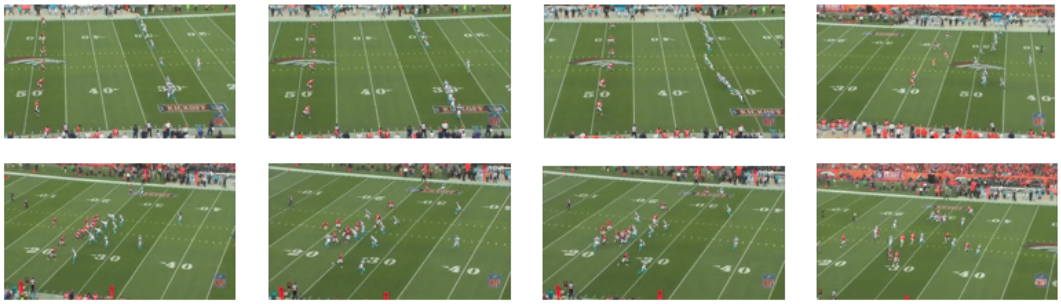

Figure 1.3: Sample input for playtype classification

map falls on a person performing an activity and the activity is correctly predicted, then we have the case that the correct prediction was made for the right reason. On the other hand, if the attention map falls on the background, then we could have that the correct prediction was made for the wrong reason. Analyzing these and similar cases would provide insights into how the network works.

For providing explanations, we use the Excitation Backpropagation algorithm (E-BP) for the network's predictions. To estimate attention maps we use E-BP which computes the neural activations in the predictor network in a top-down manner. The attention maps of all frames of all videos in the NFL football dataset provide new data that we analyze for providing explanations of the network's decisions.

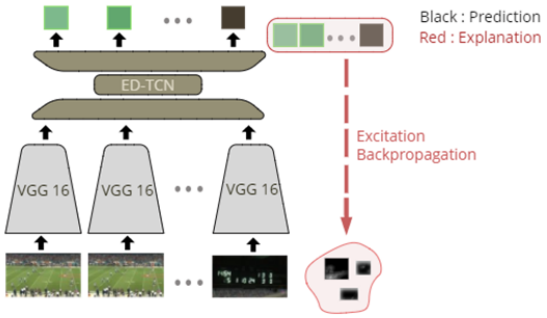

Figure 1.4: Overview of the Excitation Backpropagation method. A prediction such as 'Side View' (green) or 'Scoreboard' (brown) produces an attention map that shows the parts of the image relevant to the prediction. By using E-BP over a video segment, we obtain the attention maps for the sequence of frames in that segment.

Figure 1.5 shows our X-VAE architecture used for providing explanations to network predictions. Although attention maps relate to single predictions, we want to extract frequent patterns when considering all attention maps, as these patterns are typically meaningful. Overall, we cluster the attention maps then visually inspect the meaning of each cluster in the football domain and use these semantic words to explain the network's prediction on a new video. Formally, our explanation of a prediction is specified as a histogram of the dictionary of words representing clusters of attention maps.

For clustering we account for both spatial and temporal properties of attention maps. This is realized as a cascade of two auto-encoders where the first encodes spatial properties of each attention map and the second one accounts for temporal properties in a sequence of attention maps. In this way we project input attention maps to a latent deep feature space that captures both spatial and temporal properties of attention maps.

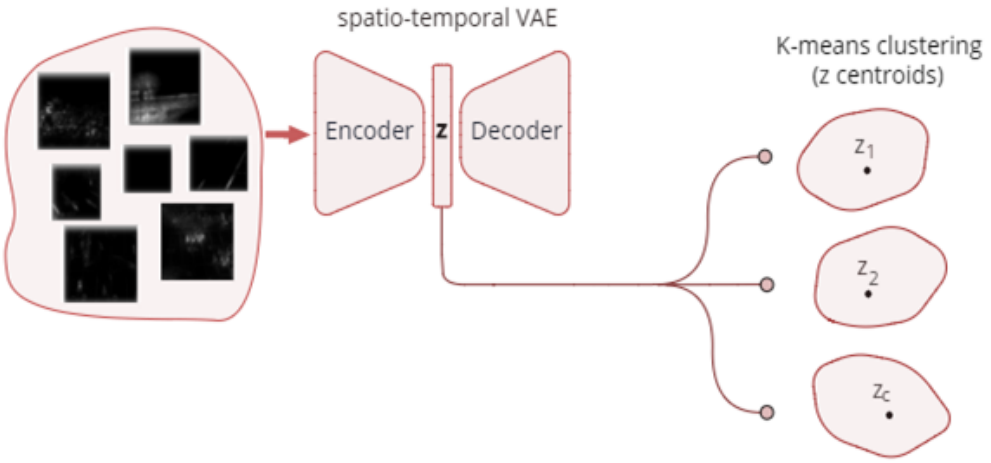

Figure 1.5: Overview of the X-VAE Architecture

Given a video, we predict its playtype segments, then compute attention maps for every frame, and assign codewords to each attention map. Then, we compute a histogram of attention map codewords for each video segment.

Chapter 2: Literature Review

In this chapter we give an overview of prior work in football analysis, followed by work in action segmentation and video classification. We also cover visual attention models. Other types of literature will be reviewed later in the text.

2.1 Football Analysis

Past work has addressed the analysis of football videos [12][3][6][7]. These attempts focus on particular parts of football analysis, like moment of snap detection, or playtype recognition. In [12], moment of snap detection is achieved by leveraging optical flow and an active cell approach that splits every video frame into cells. Hough transforms are employed in [3] to extract field lines and use mid-level feature detectors to predict playtypes. In [6], recognition of team formation is automated for further use in higher-level analysis. Other work [7] specifically targets playtype recognition but depends on player trajectories as input to the model. Our work differs from the above in that we do not leverage traditional image processing techniques with hand-crafted features, instead opting for deep learning methods where feature extraction is learned. We are not aware of any previous work for football analysis based on deep learning methods.

2.2 Action Segmentation

For action segmentation, current approaches typically extract per-frame deep features, and then use a temporal model for frame labeling. For example, in [19] the authors use an LSTM network to model feature dependencies over a fixed temporal interval. In [17] a multi-stream bi-directional recurrent neural network is presented for the task of fine-grained action detection. Well-known limitations of recurrent models include difficulty of training [14], and limited attention span [17]. We instead use [10] for temporal viewpoint segmentation, which compares favorably to the aforementioned work on bi-directional

LSTM baselines while being easy to train and use with the Excitation Backpropagation algorithm.

2.3 Video Classification

The main approaches for video classification are typically the following:

Using a convolutional network to extract features, and passing them through a recurrent neural network, as mentioned in the previous section on action segmentation.

A 3d-convolution where kernels have a receptive field that extends beyond the 2-dimensional surface of an image into a 3d volume that extends in width, height and time. As the authors note in [18], the main drawback of this method is the increased amount of GPU memory required, so that model parallelism is required to train their proposed model.

Extraction of features using a convolutional network, but passing them through an RNN at a later time, without joint training. This approach is identical to the approach mentioned in a) if the convolutional network features are pre-computed without back-progating gradients to it from the RNN network. This approach is conceptually simpler and the corresponding models are easier to train. However, the second part of our approach requires back-propagation of signals from predictions to input, which is not possible with two disjoint networks.

Combining any of the above methods with optical flow estimation can lead to better model performance, since it encodes useful motion information between pairs of frames and produces a vector field that can be used to augment the inputs in a deep network. Networks like Flownet [4][5] are trained on synthetic datasets (e.g. the flying chairs dataset) and then used on different datasets where motion can be generalized effectively. For our particular task of NFL playtype classification, such methods are not very reliable because of fast changes in camera angles, large motions of objects etc.

2.4 Visual Attention Models

There is limited prior work on the estimation of various forms of attention in deep models. The ability to find where the prediction-causing signals occur in the input of a model has been explored in many works. One of the earliest works to appear towards

this goal [20] used de-convolutional networks that were appended to directly map feature activations to the pixel space. Using feedback networks, the authors of [2] are able to infer the activation of hidden layer neurons. More recently, the authors of [21] use a modification of the gradient back-propagation algorithm to produce a distribution of pixels most likely responsible for a model's prediction. They introduce the idea of using contrastive signals to separate important parts of an image conditioned on a specific output. The contrastive version of this algorithm , called Excitation Back-propagation (E-BP), is able to produce attention maps for multiple object categories within a given image with great detail compared to other methods. We use E-BP to extract our dataset of attention maps that form the basis for analyzing explanations using our framework. E-BP has appeared in even more recent work, as in [15]. Our approach is different from [15] as we analyze an entire dataset of attention maps from the first layer of a network, instead of using explanation modules that use E-BP locally in different layers throughout the network.

Chapter 3: Playtype Recognition

3.1 Viewpoint Segmentation

The overview of our approach has been shown in Figure 1.1. The first building block of our approach is the VGG classifier [16] as illustrated in Figure 3.1. This model is used for extracting deep features from every frame in the video. These features are later used as input to the ED-TCN model for video segmentation.

The main idea of the VGG network, as compared to previous work, was to use 3×3 convolutional kernels thereby reducing the number of parameters, while also being able to increase the depth of the model significantly. Fully connected layers in the model map kernel activations to the final classes. VGG is a common feature extractor in the community.

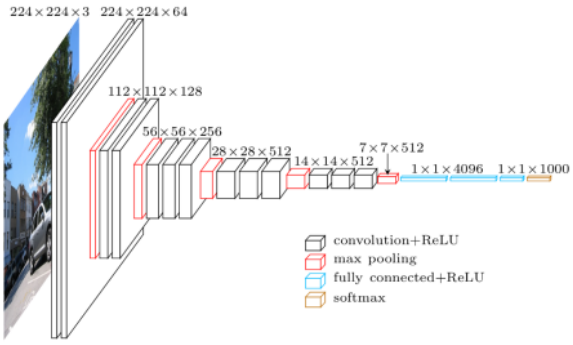

Figure 3.1: Standard VGG-16 architecture

We train the VGG model to make per-frame predictions of viewpoints. Specifically, we use a standard VGG-16 model trained on the Imagenet dataset, and remove the last fully connected layer, since the original model is trained for predictions over 1000 classes. We then append a new fully connected layer and use a softmax layer to make predictions over the three viewpoint classes. We use a learning rate of $1e-4$ and weight decay $1e-6$ as the hyper-parameters of the Adam optimizer [8]. During training, we fine-tune our

hyper-parameters using grid-search until training converges. We train our model for ten epochs as a frame-wise baseline.

Model predictions can be inconsistent in labeling similar frames under the same class because of input noise. To smooth out the predictions as well as integrate information from the whole sequence of frames, we additionally use the encoder-decoder temporal-convolutional-network (ED-TCN) on top of the VGG deep features. We again remove the last fully connected layer of the VGG model and thus are able to extract 4096-dimensional features per frame for further processing. We then use the ED-TCN model, described in "Temporal Convolutional Networks for Action Segmentation and Detection" [10] on multiple frames. Convolutional models like ED-TCN that convolve with kernels through time have surpassed bidirectional LSTM baselines. They are more appropriate for use with methods like excitation backpropagation, because they allow calculation of back-propagating signals through time while evaluating backwards passes for computing gradients only through convolutional, fully-connected, max-pooling and up-sampling layers. The original algorithm for Excitation Back-Propagation applies to these layers, but cannot be directly applied to RNNs [21].

For ED-TCN, we use two encoding layers and two decoding layers, with kernel sizes 64 and 96. We follow the suggestions of the authors and use a convolutional layer, followed by dropout, ReLU and max-pooling layers for every encoding layer in the network. For decoding layers, we invert the order of the layers, and substitute the max-pooling layer with a bilinear up-sampling layer. We use a frame window of 7 frames for the length of the convolutions through time, and predict one out of three classes per frame at the output using a fully connected layer from the convolutional features to the number of outputs. We use the same hyper-parameters as before to jointly train the VGG model with the ED-TCN model, on fixed-size sequences of frames.

3.2 Playtype Classification

Given video segments of viewpoints, we classify each segment using the *VGG+ED-TCN* model. After we have segmented the input video into viewpoint intervals, we remove all scoreboard views and separate the remaining intervals into a set of plays. The entire model uses the same VGG feature extractor as used in viewpoint segmentation to extract per-frame features, that will be used with the encoder part of ED-TCN to predict the

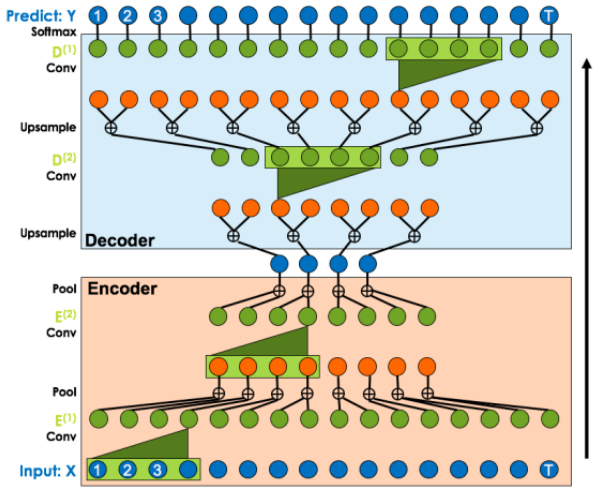

Figure 3.2: Encoder-Decoder Temporal-Convolutional-Networks. The input to ED-TCN is a set of video features from the VGG network extracted from every frame in a video sequence. In the output of the model we obtain a class label for each frame.

correct playtype. We use the same ED-TCN architecture as before. The intermediate temporal features at the layer prior to the decoder part of the network are used to make a prediction. To do so, we use two fully connected layers from the temporal features to the final playtype prediction, with the number of nodes per layer being $\{96 \times 32, 16, 3\}$. The softmax over the three outputs after the fully connected layers provides the network's playtype prediction for the entire sequence.

The inputs to the model explained above are obtained from fixed-size sequences of frames, extracted from viewpoint segments where a play occurs. During training, we predict the whole segment's playtype label from a randomly sampled contiguous sequence of frames inside the segment. During inference, a majority voting scheme allows us to predict a single playtype label for the entire segment. This is done by using a scanning window along time, where windows can be overlapping or consecutive. The incremental step for every overlapping window is one tenth of the window size.

Chapter 4: Datasets

The 2016 NFL videos are RGB videos pre-formatted to follow a 'scoreboard'-'play'-'replay' layout, without intermissions. The post-processed inputs to our models are video sequences extracted from these videos, and resized to a resolution of 256×256. Playtype annotations exist for all 2016 NFL videos but they are associated with game time instead of video time. We randomly selected a subset of 14 videos for which we manually annotated the viewpoint segments, leading to a one-to-one correspondence between the annotated playtypes and the viewpoint intervals between scoreboard views. This subset of videos contains 2149 plays for a total of 21 hours of football. We include both the play and replay information available in the videos to predict the playtype of each viewpoint interval between scoreboard views. The inclusion of replays is important because while they show the same events on the field, a different camera angle is used. Our dataset does not allow matching of viewpoints between plays and replays because their durations and number of frames per view are typically very different. This provides additional visual information for our video analysis. Our viewpoint segmentation model was trained with an additional subset of scoreboard views, used to balance the dataset.

We have some insight the challenges of segmenting viewpoints when we try to create an automated viewpoint annotation architecture which was not based on deep learning models. We implemented a method based on color histogram comparisons. We divided every image frame into a grid of size 4×4, and calculated the correlation between the color histograms of every frame in the video as it relates to the average of the scoreboard frames. For some videos, the median and minimum values of the correlations of all image parts of the grid were useful in predicting whether the scoreboard was present or not, however field logos as well as noisy frames made the task of selecting a clear dividing point between field views and scoreboard views difficult. Specifically, for varying thresholds of hyper-parameters that divided the frames into two sets, we either obtain a consistent number of plays for most thresholds, or fail to match the number of plays with the number of pre-existing playtype annotations. Thus only a very small number of videos could be automatically annotated.

After using contrastive Excitation Backpropagation on viewpoint and playtype model predictions, we obtained two additional datasets of attention maps, one for viewpoint sequences and another one for playtype sequences. In the training set of attention maps we only include the attention maps where the network makes correct predictions.

Chapter 5: Experiments of Video Analysis

We evaluate viewpoint segmentation predictions using **frame-wise accuracy**, **confusion matrices**, as well as textbfIoU scores and **Precision-Recall curves** over the segmented viewpoint intervals. Frame-wise accuracy is not enough to determine whether the intervals we produce are of high enough quality to use with playtype classification. Errors in single frame prediction may not contribute enough to frame-wise accuracy to be deemed important, however single frame errors leading to erroneous viewpoint interval detection will lead to entire segments of viewpoints being misclassified by our playtype classification model.

For playtype classification we note that the intervals produced by viewpoint segmentation using our best model are identical to the ground truth intervals, with minor differences only in the exact location of the beginning and ending frames of each viewpoint interval. Thus our evaluation of playtype classification focuses on the differences of prediction accuracy when using the predicted intervals instead of the ground truth intervals. In both cases, the differences in the input intervals lead to insignificant differences between prediction accuracy, as discussed in the playtype experiments section below.

Our baseline model for viewpoint segmentation is a **frame-wise** model that predicts a class for every input frame without using any temporal information. The next models use ED-TCN over fixed window intervals of 32 or 196 frames, called **EDTCN-32** and **EDTCN-196** respectively. Finally, the model **ED-TCN (no fine-tuning)**, where the initial frame-wise VGG model has been trained for 10 epochs but we only update the ED-TCN parameters after this initial training phase. This model uses window intervals of 32 frames.

For playtype classification, we first discuss our ablation studies over multiple model architectures, regarding the use of EDTCN. Our baseline model uses the decoder part of the network, includes the final VGG model trained on the task of viewpoint segmentation, and includes a window size of 32 frames. For determining the best architecture regarding EDTCN, we exclude the 'Other' playtype category so that the task of playtype

classification is a binary classification task. We refer to the validation set of our dataset as a testing set during the comparisons with other models. We include comparative evaluations by removing the decoder of the EDTCN model, training from scratch, as well as expanding the window size to 64 frames.

We then evaluate multiple models on the task of playtype classification over three playtype categories. Our baseline for these experiments is a model that is trained from scratch without a decoder network. We compare models that use 8 frames, 16 frames and 32 frames. We then turn our attention to the importance of including both side view and front view of plays, by comparing the performance of models over single views of the field, or a combined view of both. Our final results on the testing set show confusion matrices and final testing accuracy on both ground truth and predicted intervals. For this final evaluation, we use a window size of 32 frames on both views of the play.

Our first experiments focus on viewpoint segmentation and playtype classification, using the dataset of 2016 NFL videos as introduced in Chapter 4. We achieve a 99.5% testing accuracy for viewpoint segmentation. By classifying the previously segmented video intervals by playtype, we achieve 80.1% testing accuracy. We also evaluate our models using IoU scores and Precision-Recall curves for a performance measure over the entire pipeline of viewpoint segmentation and playtype classification.

5.1 Viewpoint experiments

Our playtype recognition architecture consists of two modules, a viewpoint segmentation model and a playtype classification model. The first module is comprised of the VGG network, that has been pre-trained on isolated frames, and the ED-TCN network that uses the VGG features on every frame to segment the video into plays using convolutions through time. We split the dataset into training and testing videos, using an 80%-20% split. For the training videos, we remove 10% of the frame sequences to use for validation and the rest are used for training. The testing videos come from stadiums and games that the models have not seen, which is suitable for thoroughly testing the generalization ability of the models. We train the baseline *framewise* model for viewpoint segmentation, which we will use as pre-training for the combined *VGG+ED-TCN* architectures.

We then evaluate the viewpoint segmentation models with a per-frame accuracy metric. We include the pre-trained *framewise* model as well as three models which em-

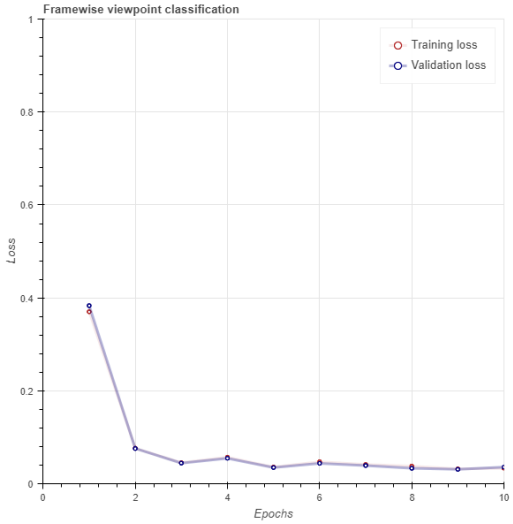

Figure 5.1: Viewpoint classification / Loss

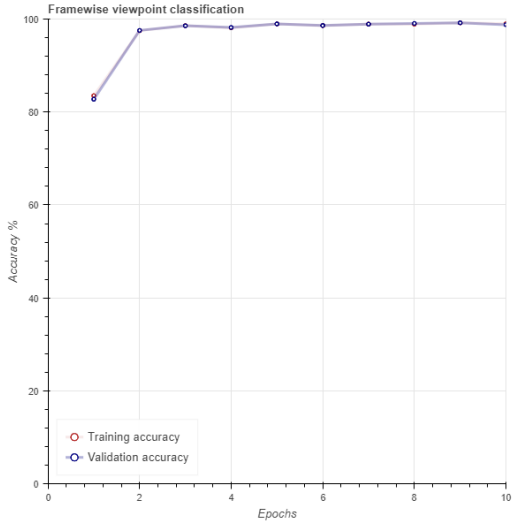

Figure 5.2: Viewpoint classification / per-frame accuracy

P \ G.T.	Front View	Side View	Scoreboard
Front View	0.487	0.003	0.001
Side View	0.001	0.420	0.001
Scoreboard	0.001	0.000	0.085

Figure 5.3: Training set normalized confusion matrix over the viewpoint classes. Predicted vs. ground truth classes.

P \ G.T.	Front View	Side View	Scoreboard
Front View	0.485	0.003	0.000
Side View	0.001	0.417	0.001
Scoreboard	0.001	0.000	0.091

Figure 5.4: Testing set normalized confusion matrix over the viewpoint classes. Predicted vs. ground truth classes.

ploy ED-TCN over windows of frames, namely, *EDTCN-196* with a window size of 196 frames, and *EDTCN-32* with a window size of 32 frames as well as *EDTCN-32 (no fine-tuning)* where the VGG network's parameters are not updated. We evaluate on both consecutive frame sequences and sliding-window overlapping frame sequences, for both validation and testing sets.

We also use the Intersection-over-Union (IoU) metric to determine whether the segmented temporal intervals overlap sufficiently with the ground truth intervals. We employ the Viterbi algorithm over the resulting intervals to smooth-out any inconsistencies that would lead to invalid viewpoint changes that never occur in the dataset. The pa-

	Training Accuracy	Validation Accuracy (consecutive)	Validation Accuracy (sliding)	Testing Accuracy (consecutive)	Testing Accuracy (sliding)
framewise	0.989	0.988	0.988	0.987	0.987
EDTCN-32	0.983	0.989	0.989	0.988	0.989
EDTCN-196	0.984	0.987	0.987	0.984	0.984
EDTCN-32 (no fine-tuning)	0.994	0.995	0.995	0.994	0.995

Figure 5.5: Viewpoint segmentation per-frame accuracy

rameters for the Viterbi algorithm are obtained from the statistics of the training dataset.

	front view	side view	scoreboard	all classes
framewise	0.967	0.985	0.978	0.977
EDTCN-32	0.976	0.981	0.968	0.975
EDTCN-196	0.952	0.949	0.848	0.917
EDTCN (no fine-tuning)	0.988	0.991	0.967	0.981

Figure 5.6: Viewpoint segmentation IoU before viterbi

	front view	side view	scoreboard	all classes
framewise	0.979	0.981	0.991	0.983
EDTCN-32	0.981	0.981	0.971	0.978
EDTCN-196	0.952	0.956	0.852	0.921
EDTCN (no fine-tuning)	0.989	0.991	0.968	0.982

Figure 5.7: Viewpoint segmentation IoU after viterbi

Viewpoint segmentation needs to produce high-quality intervals, in the sense that the two end-points of each interval should be as close to the ground truth end-points as possible. To evaluate the quality of viewpoint segmentation, we use Precision-Recall curves over multiple IoU thresholds in the range $[0.5, 0.9]$, as shown in Figures 5.8 and 5.9 respectively.

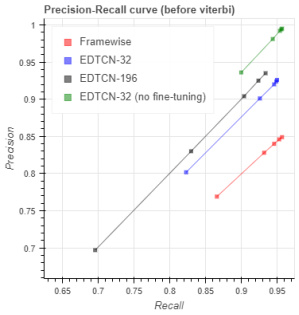

Figure 5.8: Precision-Recall curve for all viewpoint models for different IoU thresholds (before viterbi).

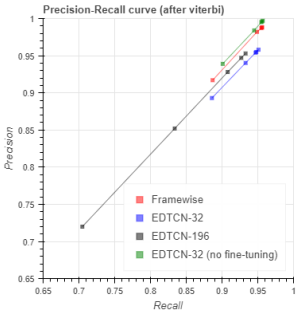

Figure 5.9: Precision-Recall curve for all viewpoint models for different IoU thresholds (after viterbi).

5.2 Playtype experiments

We first attempt to use the same architecture used for viewpoint segmentation with multiple adjustments for playtype recognition. We compare models with 32 uniformly sampled frames in the input, where the output of ED-TCN consists of 16-dimensional features that are used as input to the fully connected layers leading to the playtype prediction. We include a model with 64 uniformly sampled frames as input, and also evaluate performance with lack of pre-training for the VGG, which is trained from scratch on the playtypes. We remove the decoder network as we are no longer interested in per-frame labeling, but predicting a single label for the entire duration of the play. We see that the performance of the model when no decoder network is used outperforms all other models while including less parameters. This model is used for the second set of experiments for playtype classification.

We now provide details over the experiments on the playtype classification. As explained in the first part of the experiments of video analysis, we evaluated multiple models for our ablation studies. Our results show the performance of the model when we include more frames per play, remove the decoder network (so that there is no action segmentation) and initialize without loading a pre-trained viewpoint VGG model. For our initial set of experiments, we only train and evaluate our models on binary classification of the two majority classes. The test set used for this set of experiments is the validation set of the next set of experiments.

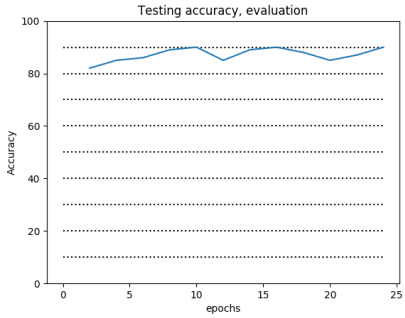

Figure 5.10: Testing accuracy, baseline model

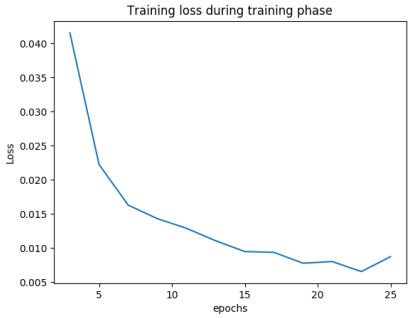

Figure 5.11: Training loss, baseline model

To determine whether the playtype classification would show improvement if more frames were included, we run experiments with more frames per play (which are still uniformly sampled from the frames of the play).

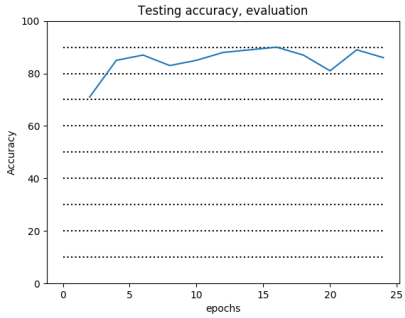

Figure 5.12: Testing accuracy, model has 64 frames

Figure 5.13: Training loss, model has 64 frames

Removing the decoder part of the network means that we are only convolving through time and using less parameters. This is a more natural approach to classification of an entire play with a single label, so it is worthwhile to note whether the accuracy of the model is lower when we do not use a decoder network.

Removing the pre-trained VGG model induces better attention maps but the testing accuracy does not drop compared to other models.

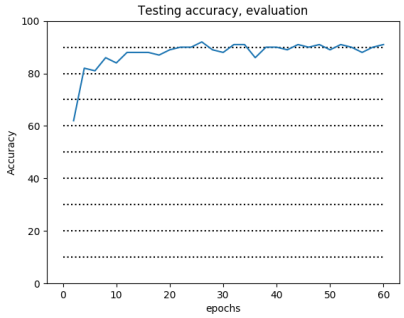

Figure 5.14: Testing accuracy, model has no decoder

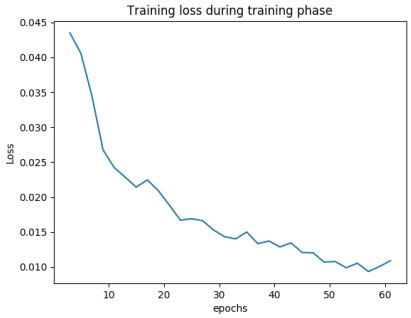

Figure 5.15: Training loss, model has no decoder

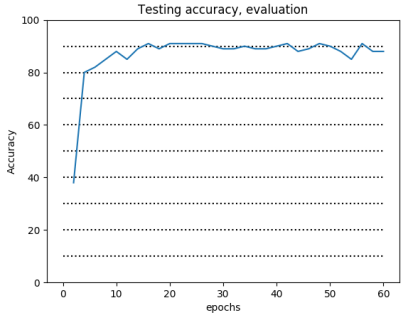

Figure 5.16: Testing accuracy, model without pre-training

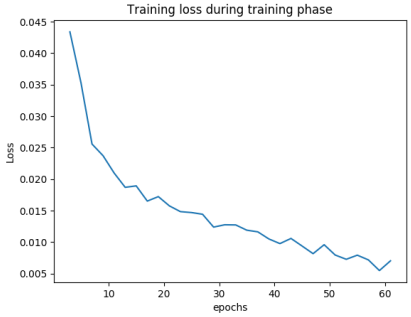

Figure 5.17: Training loss, model without pretraining

Model	Test acc. (after model converges)
baseline model	90%
64-frames	87%
no-decoder	92%
no-pretraining	88%

In conclusion, the best models for the task of playtype classification work regardless of pre-training or inclusion of the decoder network. Thus the model used for the clustering analysis will be a model trained from scratch without the decoder part of ED-TCN.

We now train our playtype classifier on three classes, using only the encoder network, and classifying the playtype afterwards. We train the network from scratch, and then

freeze the VGG layers to only train the temporal part of the network. We use sliding windows and consecutive windows for evaluation. We obtain 85.4% on the validation set for 3 playtypes with the final model, and 80.6% on the testing set. Testing set frames and plays are outside the training dataset and involve new unseen stadiums.

	Val acc. (epoch 10)	Val. acc (epoch 20)
8-frames	58%	60%
16-frames	72%	80%
32-frames	83%	83%

Figure 5.18: Comparison of different window sizes

	Val acc. (epoch 10)	Val acc. (epoch 20)
front viewpoint	67%	65%
side viewpoint	65%	63%
both viewpoints	81%	83%

Figure 5.19: Comparison of different viewpoint inclusions

As shown in Figures 5.18 and 5.19, we tested different window sizes and viewpoints, with early stopping. At epoch 20, it is clear that both viewpoints with a window size of 32 frames is the most appropriate.

Over-fitting is a concern when combining the VGG model with the ED-TCN model, so we freeze the model of VGG after epoch 20 and train for another 20 epochs we only update the ED-TCN model parameters. We evaluate on overlapping and non-overlapping intervals, but all predictions remain identical. We also find that when the viewpoint segmentation is used to detect scoreboard vs. field views (which include front and side views), 100% of ground truth field viewpoints are included for playtype classification.

Table 5.1: Predicted intervals, confusion matrix

Classes	'Rush'	'Pass'	'Other'
'Rush'	0.19	0.06	0.02
'Pass'	0.03	0.38	0.01
'Other'	0.03	0.06	0.23

Table 5.2: Ground truth intervals, confusion matrix

Classes	'Rush'	'Pass'	'Other'
'Rush'	0.19	0.06	0.01
'Pass'	0.03	0.39	0.01
'Other'	0.03	0.05	0.23

Table 5.3: Predicted intervals, accuracy

Classes	Test acc. on predicted intervals
all classes	80.6%
'Rush'	74.7%
'Pass'	90.9%
'Other'	76.1%

Table 5.4: Ground truh intervals, accuracy

Classes	Test acc. on ground truth intervals
all classes	81.7%
'Rush'	74.6%
'Pass'	92.6%
'Other'	77.8%

At inference time, we compare ground truth predictions with viewpoint segmentation predictions where all intervals are assigned the majority playtype they overlap with. We see a slight decrease in performance when using the predicted intervals. These are the final results over the selected model that we determined was most appropriate given our ablation studies.

Chapter 6: Visual Explanations

6.1 Excitation Backpropagation

To understand model predictions, we use the contrastive Excitation Backpropagation method. For every input-prediction pair of either the viewpoint segmentation model or the playtype classification model, we can back-propagate gradients that produce gray-scale maps of attention. These attention maps relate the prediction at the output of the model to the input signals that have caused the prediction.

Excitation Backpropagation is based on a winner-takes-all (WTA) formulation for determining which neurons of a neural model have the highest likelihood for responsible for the above layer's activations.

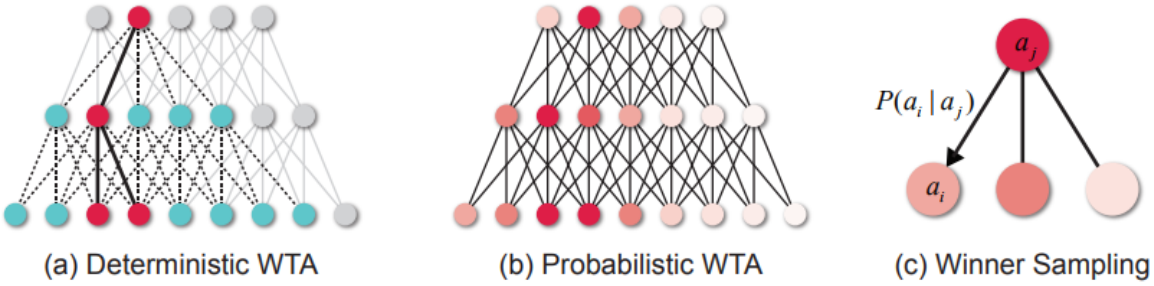

Figure 6.1: Excitation Backpropagation, probabilistic WTA formulation

Back-propagation of signals can be defined in a recursive way. For any two layers, the positive-weight connections between neurons relate the activations of neurons at both ends of a connection to determine the probability that the bottom-layer layer neuron will activate. By applying the same process repeatedly in a top-down manner, we obtain the probability distribution of neural contribution at any layer.

$$P(a_j) = \sum_{a_i \in P_j} P(a_i|a_j)P(a_j)$$

$$P(a_i|a_j) = \left\{ \begin{array}{l} Z_i a_j w_{ij}, \text{ if } w_{ji} \geq 0 \\ 0, \text{ otherwise} \end{array} \right\}$$

We use Excitation Backpropagation until we reach the first layer of the VGG model, to obtain detailed attention maps that are as close to the dimensions of the input images as possible. Excitation Backpropagation is used through both the VGG and ED-TCN models, and it produces attention maps for multiple frames simultaneously.

6.2 Spatio-temporal Autoencoders

Attention maps are the contributions of pixels in the original frame towards a single model prediction. These attention maps exist in isolation, so the next part of our approach involves grouping attention maps that have common patterns as typically these patterns can be meaningful in the context of predictions. In order to relate common patterns in attention maps, we need to resolve two problems. The first one is translation invariance. When an attention map includes a pattern that occurs throughout many attention maps at different location, we want to be able to represent it in a similar way. The second one is using a better representation compared to raw pixel space, so that we can capture more high-level features of the attention maps. Such representations can be obtained from the latent dimension of an auto-encoder, in absence of any supervised information over the attention maps.

To capture both translation invariance as well as obtain a compact representation of attention maps that is appropriate for groping similar attention maps together, we use a convolutional variational auto-encoder (Conv-VAE).

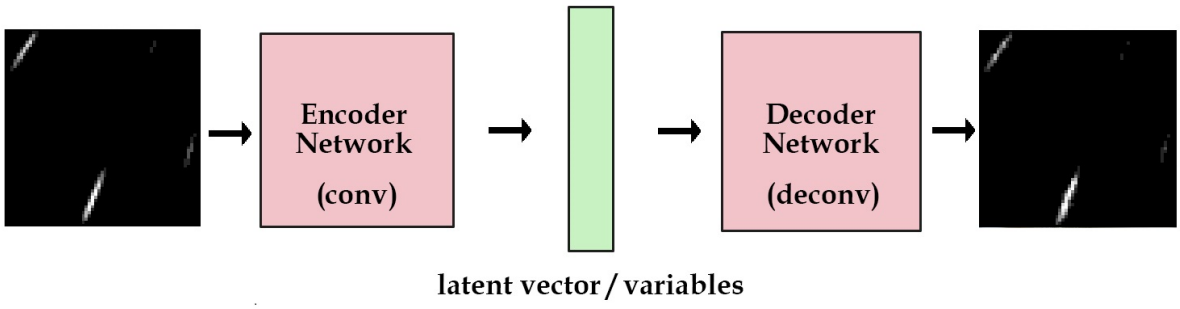

Figure 6.2: Convolutional variational auto-encoder

Our Conv-VAE model consists of four alternating convolutional and max-pooling layers for the encoding network. We use 96 convolutional kernels for the first two convolutional layers and 196 kernels for the next two convolutional layers. Three fully connected layers map the convolutional features to the latent code, which has dimension 32. The two hidden layers have 512 hidden nodes each. We construct the decoder network by reversing the ordering of the layers, and substituting max-pooling and convolutional layers with bilinear-upsampling and transposed convolutional layers respectively.

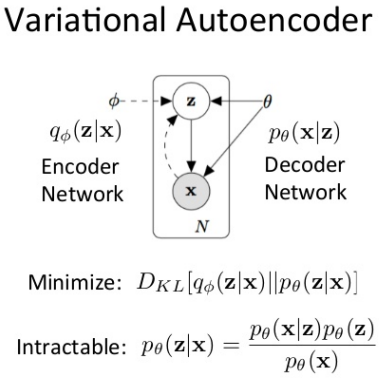

Figure 6.3: Variational Autoencoder, original formulation of the architecture

To train the auto-encoder, we use the standard loss for variational auto-encoders, which is the sum of two loss functions. The first is a reconstruction loss, where mean squared error is used to determine if the output of the auto-encoder is pixel-wise similar to the input. The second one is a regularization loss, KullbackLeibler divergence (KL-divergence), which is a measure of how the latent distribution of z matches a prior distribution, in this case a multivariate unit Gaussian.

$$Total\ VAE\ Loss = \sum_{i=1}^{N} l_i$$

$$l_i(\theta, \phi) = -E_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i|z)] + KL(q_\theta(z|x_i)||p(z))$$

Our intent is to cluster the latent representations of all attention maps to groups we call code-words. These code-words can be used to explain the way a model makes predictions. By borrowing from Natural Language Processing, we aim at connecting the latent representations of attention maps with the context in which an they occur. Word-2-Vec [13] models are appropriate in the context of word embeddings, in order to represent semantically similar words closer together in an embedded space. However, since the original attention maps as well as their corresponding latent code is continuous, we cannot use one-hot encodings of either the attention maps or their latent representations that Word-2-Vec models require.

To resolve this problem, we use a conditional variational auto-encoder that produces a generative distribution of the context of each attention map in the dataset. This resembles the skip-gram architecture of Word-2-Vec models, where an input word is used to predict the context. For a given sequence of attention maps and a window centered on any part of the sequence, we first use the spatial auto-encoder to obtain the latent representations of the attention maps within the window. Then we train the conditional variational auto-encoder to predict the latent representations of nearby attention maps conditioned on the attention map in the center of the window.

The conditional variational auto-encoder does not use convolutional layers, instead employing two fully connected layers for both the encoder and decoder network. We use a latent space of dimension 4 for predicting the context of the conditioned latent spatial representation of our input frame. Both the context being reconstructed as well as the conditioned input are the spatial latent codes that our Conv-VAE produced. To train

the conditional variational auto-encoder, we follow the same procedure as for the Conv-VAE model, except that both input and latent representation are concatenated with the conditional latent code of the input frame. The entire model of the two variational auto-encoders is jointly trained end-to-end.

6.3 Clustering

Given a sequence of attention maps, we want to obtain a contextual representation of each map. To do so, we concatenate the latent representations of the conditional variational auto-encoder as it reconstructs multiple contextual samples around our current attention map. We then concatenate its own spatial latent representation to the conditional representations, to form a spatio-temporal representation for the attention map.

After obtaining the representations for all attention maps, we perform a k-means clustering procedure to produce groups of semantically similar attention maps. To provide an explanation after the clustering, we first find cluster membership for all attention maps, after extracting their spatio-temporal representation. We then provide the representatives of each cluster as exemplars for the type of reasoning that is related to our current attention map.

6.4 Visual inspection for finding semantic meaning

A user can assign semantic meaning to each cluster that is considered meaningful through visual inspection by associating each cluster with an English word. These verbal meanings can then be associated with the clusters, along with the visual representatives closest to each cluster center. This way each sequence of codewords corresponds to list of meaningful English words and attention map parts.

We use watershed segmentation over the attention maps closest to the clustering centers to obtain map parts which explain the type of cluster the attention map belongs to. By separating the relevant image parts of every visual codeword, we can visualize a collection of the most significant visual cues that our model has learned to focus on, to make predictions that are semantically similar to the current one.

We visualize the presence of every meaningful cluster at every frame, quantifying the number of times each cluster is present in this sequence of frames inside a histogram of visual codewords and associated English words.

Chapter 7: Experiments of Visualizing Attention Maps

The dataset we use for our explanations is obtained by using Excitation Back-propagation over the training set of NFL football videos. We obtain one dataset for attention maps of viewpoint segmentation predictions and another dataset for attention maps of playtype classification predictions.

The first experimental section contains an overview of the type of data we collect after applying Excitation Backpropagation through time. We show results over multiple models covered in the Video Analysis chapter, for both viewpoints and playtypes.

In the next section, we discuss experiments for the qualitative evaluation of our X-VAE framework. We compare reconstructed inputs to the outputs of the variational auto-encoders over attention maps in a separate testing dataset, to show that our models have not overfit the training dataset. To do so, we use a $90\% - 10\%$ split of each dataset of attention maps into training and testing sets. We also explore how the spatial and contextual auto-encoders map attention maps to latent codes.

In the clustering section of the experiments, we discuss our clustering results. Using clustering over the attention maps as a baseline, we show why it is important to learn abstract representations of the features in attention maps using the spatial auto-encoder, and how it compares to the inclusion of contextual information. Thus we compare our clustering of raw attention maps to the clustering of spatial latent codes as well as the combination of spatial and contextual latent codes. We then associate clusters with English words. To do so, we visually inspect all the clusters, and choose the ones that prove semantically meaningful based on the original images that are associated with these clusters.

In the final section of our experiments, we apply our entire method to both viewpoint and playtype models and present the resulting visual and textual narratives associated with specific sequences of images. Specifically, we provide histograms of codewords over sequences of attention maps, and relate these codewords to the English words they have been annotated with.

7.1 Excitation Backpropagation through time

In this section we want to discuss some qualitative results over the NFL playtype classification task. We introduce pairs of sequences where the original images are compared to the attention maps produced by EB when it is applied throughout both the ED-TCN model and the VGG model.

Without a contrastive signal to remove common excitations between categories, we get signals from multiple classes over the input frames. These signals cannot be used to distinguish between classes that are present in the same image or sequence of images. A trained model will produce attention maps for all frames, and attend to multiple class signals per frame as shown in Figure 7.1.

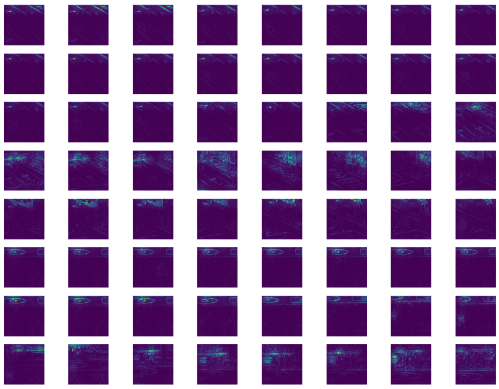

Figure 7.1: Attention maps for standard EB

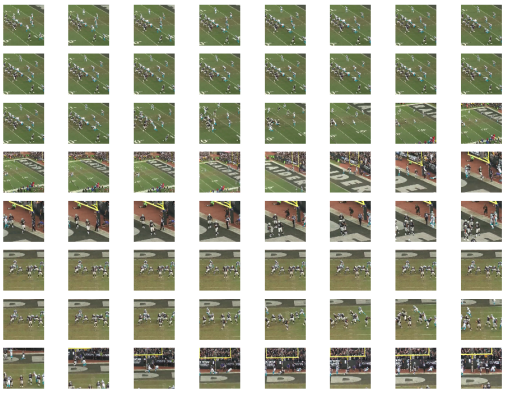

Figure 7.2: Original images for standard EB

Consequently, all further results will be shown relative to the contrastive attention maps for predictions that match the ground truth. Our initial results showed better performance over models that were not pre-trained with viewpoint annotations.

Our next comparisons reflect our model's ability to provide meaningful features per frame that can be later accumulated into a single label. By using the ED-TCN model to predict a single label for the entire sequence, we obtain the attention maps illustrated in Figures 7.3 to 7.6.

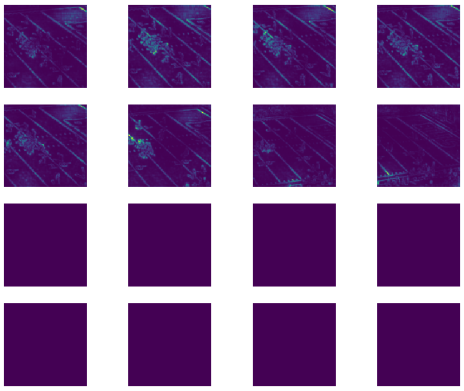

Figure 7.3: Contrastive EB, trained model

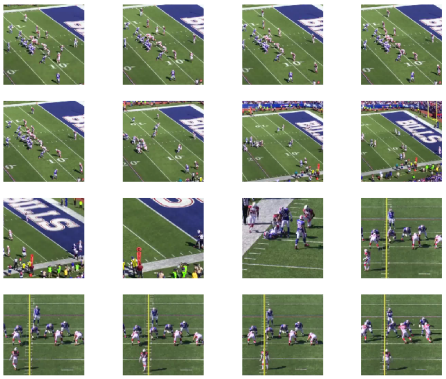

Figure 7.4: Original images, trained model

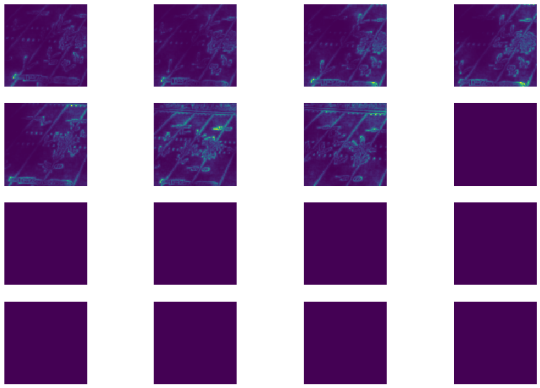

Figure 7.5: Contrastive EB, trained model

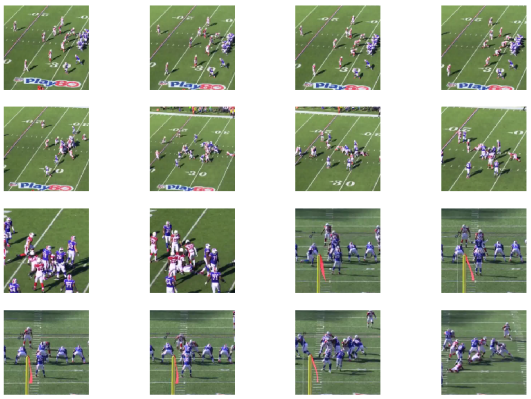

Figure 7.6: Original images, trained model

Increasing the number of frames per sequence to 64, we can see if the attention maps contain major fluctuations between frames. We notice that the attention maps are unavailable for parts of the playtype sequences for both 64 frame models and 16 frame models.

In Figures 7.9 to 7.12 we see that the most discriminative results are obtained with models that either do not use pre-training on viewpoints or do not employ decoders but

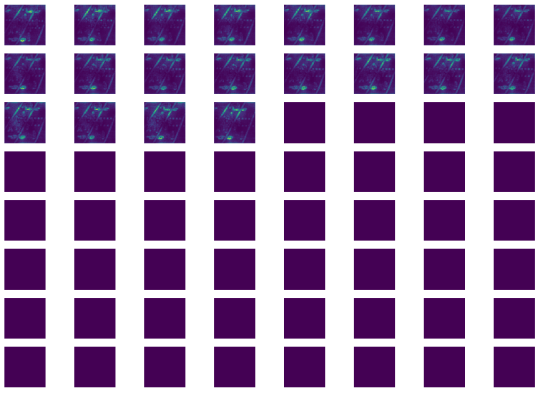

Figure 7.7: Contrastive EB, play-type model with 64 frames

Figure 7.8: Original images, play-type model with 64 frames

instead only convolve through time before making the final classification decision.

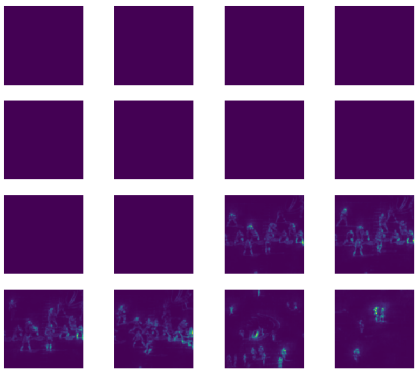

Figure 7.9: Contrastive EB, no viewpoint pre-training

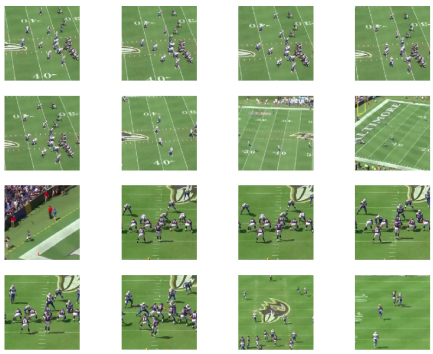

Figure 7.10: Original images, no viewpoint pre-training

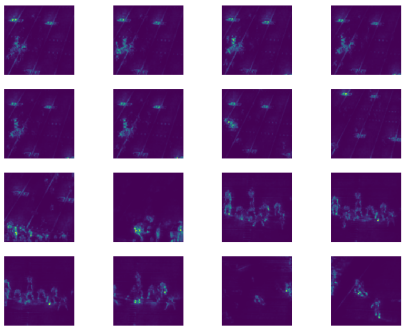

Figure 7.11: Original images, no decoder

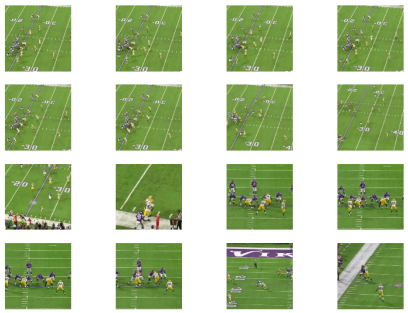

Figure 7.12: Contrastive EB, no decoder

7.2 Spatio-temporal Autoencoders

In Figures 7.13 and 7.14 we can see reconstructions of attention maps from the viewpoint segmentation task. Using the spatial variational auto-encoder, we can extract the latent representation that best reconstructs a given attention map. The 32-dimensional latent codes used for the reconstruction appear below the attention maps.

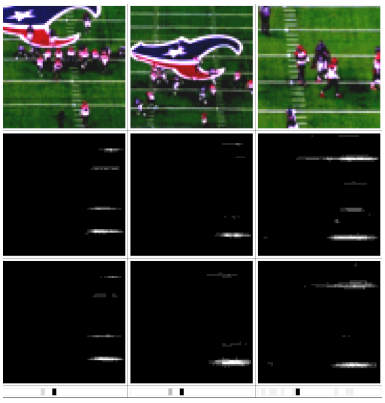

Figure 7.13: Viewpoint latent code samples. From top row to bottom row: original images, corresponding attention maps, reconstructions of attention maps, latent codes.

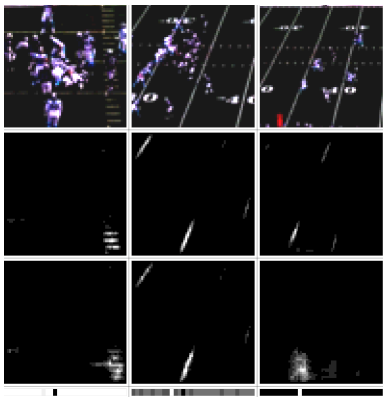

Figure 7.14: Another set of viewpoint latent code samples. The similarity between the second and third frame is apparent even though the reconstruction is not perfect.

The conditional variational auto-encoder accepts an input frame and is a generative model of its context. By randomly sampling from the conditional auto-encoder, we can obtain attention maps that are likely to occur near our input attention map. The reconstruction is over the latent space of the original spatial auto-encoder. We visualize such reconstructions as illustrated in Figures 7.15 to 7.17.

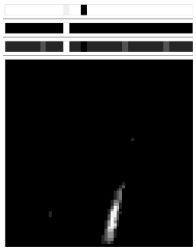

Figure 7.15: Conditional sample, the three spatial latent codes correspond to a frame before the input frame, the spatial code of the input frame and the spatial code of the sampled frame.

Figure 7.16: Another conditional sample, for the same latent codes of the prior frame and the input frame. In both these examples, the sampled latent codes are more similar to the input frame.

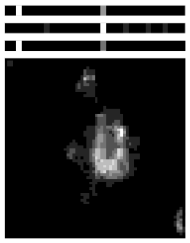

Figure 7.17: A conditional sample for the context of a frame containing a group of players, from playtype attention map sequences. Both prior, current and sampled latent codes are similar.

7.3 Clustering

For our clustering results, we use k-means clustering over the latent space of both the spatial and temporal VAEs. Given an input frame, we use the spatial VAE to obtain the latent representation of the frame. For nearby frames around the window, we obtain their latent representation as well. The temporal VAE accepts the latent code of the input frame, and tries to reconstruct the latent codes of nearby frames. By doing so, we obtain the contextual latent representations of these nearby frames, and concatenate the spatial latent code of the input frame with the temporal latent codes of the nearby frames.

We thus obtain a 64-dimensional representation of each frame that relates to its spatial configuration as well as the context in which it occurs. After clustering, we can use the example nearest to a clustering center as the codeword representative of the

entire cluster. We first visualize the results of clustering over raw frames and spatial latent codes.

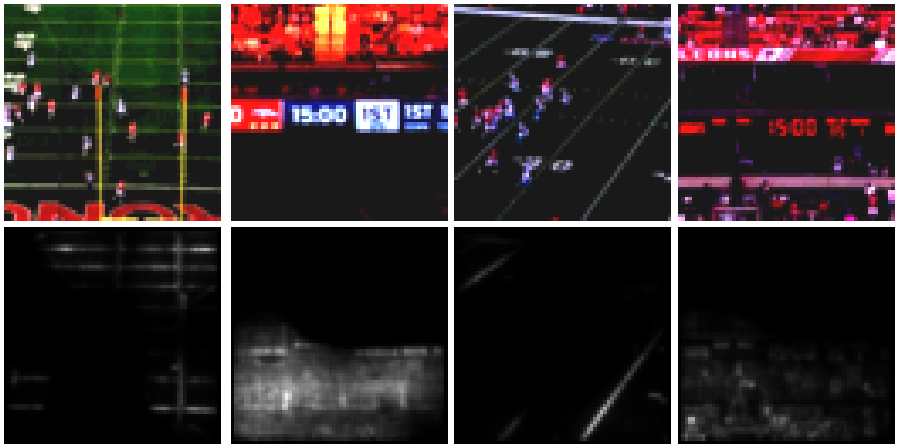

Figure 7.18: Clustering collection of attention maps for viewpoints on the raw pixel data. Clusters are typically not meaningful, as they contain multiple mixed visual categories.

Figure 7.19: Clustering collection of attention maps for viewpoints on the spatial latent codes. The clustering no longer produces clusters that confuse scoreboards with field views, but vertical and horizontal lines are mixed.

In Figure 7.20 we illustrate the clustering mechanism that assigns each input frame to a contextual frame. We extract the relevant parts of each attention map to provide

clues as to the whereabouts of important visual cues inside the frame, that best explain the model prediction.

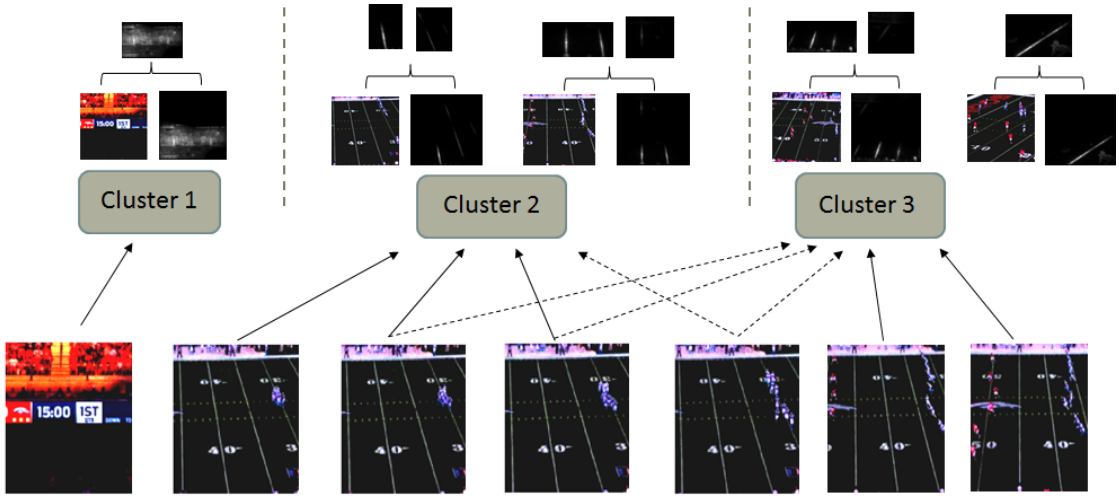

Figure 7.20: Clustering and attention frame membership on viewpoints based on spatio-temporal similarity.

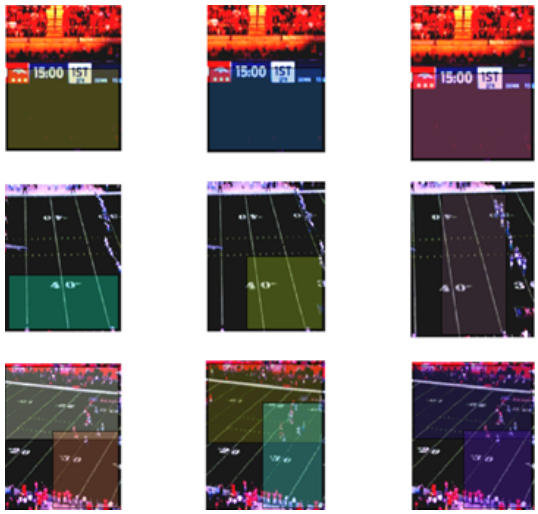

Figure 7.21: Extraction of relevant parts based on attention map segmentation over the original frames.

We also extract the important visual cues from the original frames as shown in Figure 7.21. By using watershed segmentation over the input attention maps we can overlay bounding boxes over areas that correspond to the important parts of the input frame.

To ground our visual codewords to specific terms, to be used in conjunction with the histograms of explanations, we provide an illustration of our method in Figure 7.22. We visually inspect all clusters that group semantically similar frames together, and give them a corresponding English name that represents the clustering category.

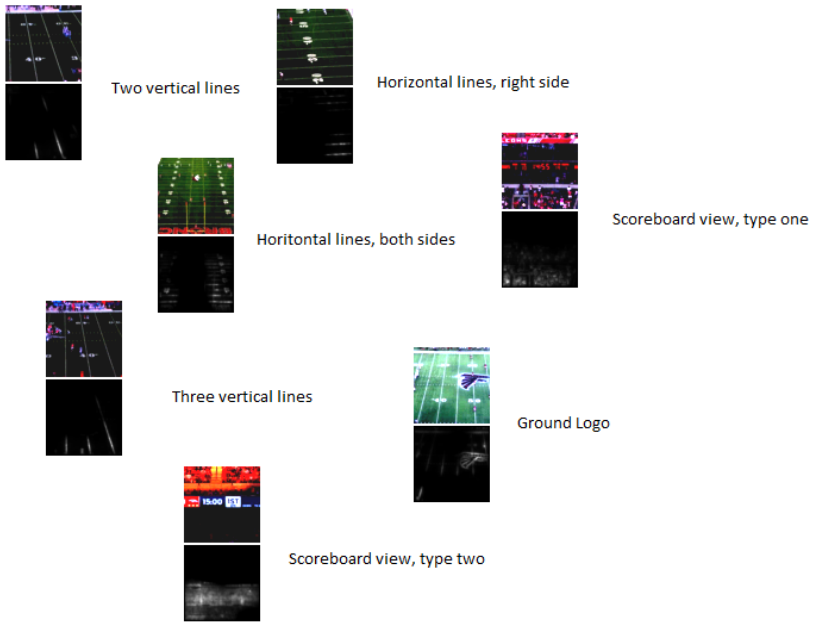

Figure 7.22: Viewpoint groups produce clustered representations from which we pick the closest exemplar. By visual inspection, we can introduce different English labels to each cluster. For the case of spatio-temporal latent code clustering, we find that most clusters are separated into meaningful groups with few outliers for both viewpoints and playtypes.

7.4 Providing Explanations

For the final explanation of a model's prediction, we provide a histogram of codewords over a sequence of images. We have shown how each codeword corresponds to a collection of visual cues that reveal where the model focuses when making a prediction. We now show that sequences of frames can be represented as a histogram of codewords with associated English names, providing a clear view of the set of visual codewords that are most relevant in the predictions made for the entire sequence.

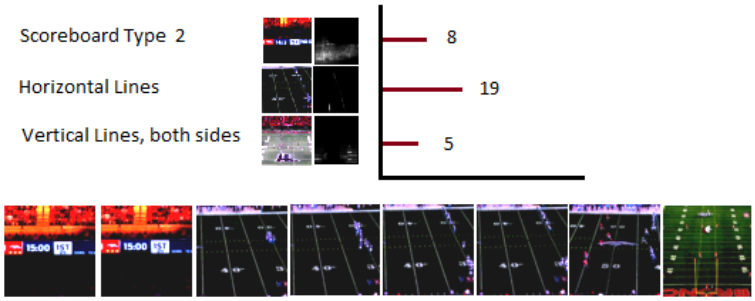

Figure 7.23: Histogram of viewpoint groups to which a sequence of frames belongs to. The sequence contains thirty two frames, from which we visualize eight. During inference, we predict the cluster each frame belongs to to produce a histogram of English codewords along with their corresponding visual representatives.

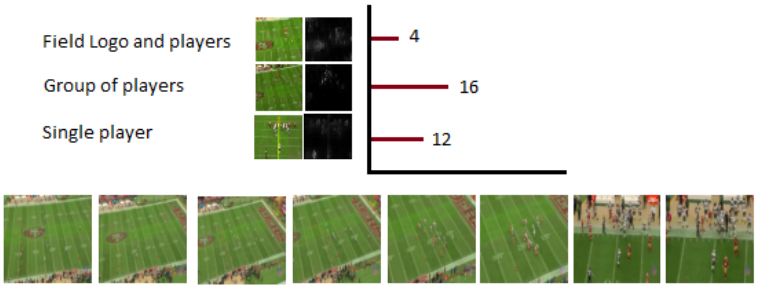

Figure 7.24: Histogram of playtype groups to which a sequence of frames belongs to. We see that the final number of clusters suffices to detect similarities in original frames even though clusters with similar attention maps come from multiple videos. Thus the representatives of the clusters come from the same video as the inference video interval.

Chapter 8: Conclusion

We have shown that Excitation Backpropagation is a very powerful tool in providing explanations for model decisions. Combined with unsupervised methods and clustering of the attention maps into semantic categories, we provide a way to produce visual explanations without human interference. Using our method, it is possible to not only provide a thorough qualitative evaluation of a model but also make explicit the representations of the reasoning leading to model predictions, shedding light into the complex interactions inside modern deep learning architectures.

For our video analysis, we get robust results for viewpoint segmentation and playtype classification, which can then be used together to form a complete playtype recognition system. Our explanatory X-VAE framework, which minimally requires the extraction of attention maps, provides a visualization of the reasoning patterns behind model predictions, and is model agnostic.

Our work has highlighted the potential for conceptual explanations of deep models that extend beyond image classification to the realm of video action recognition. We have not extended our findings into deep models that segment or localize objects in images, and have only used convolutional architectures to extract attention maps. Currently, our approach includes multiple hyper-parameters, like window sizes or latent representation sizes, that need to be hand-tuned. More sophisticated formulations can in the future allow for a system that either automatically determines the values of these parameters or entirely circumvents the need to use them.

Bibliography

[1] https://www.nfl.com/.

[2] Chunshui Cao, Xianming Liu, Yi Yang, Yinan Yu, Jiang Wang, Zilei Wang, Yongzhen Huang, Liang Wang, Chang Huang, Wei Xu, Deva Ramanan, and Thomas S. Huang. Look and think twice: Capturing top-down visual attention with feedback convolutional neural networks. *ICCV*, 2015.

[3] Sheng Chen, Zhongyuan Feng, Qingkai Lu, Behrooz Mahasseni, Trevor Fiez, Alan Fern, and Sinisa Todorovic. Play type recognition in real-world football video. *WACV*, 2014.

[4] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Husser, Caner Hazrba, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *ICCV*, 2015.

[5] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. *CVPR*, 2017.

[6] Shaunak Ahuja Karthik Muthuswamy Narendra Ahuja Indriyati Atmosukarto, Bernard Ghanem. Automatic recognition of offensive team formation in american football plays. *CVPRW*, 2013.

[7] Shaunak Ahuja Bernard Ghanem Narendra Ahuja Jagannadan Varadarajan, Indriyati Atmosukarto. A topic model approach to represent and classify american football plays. *BMVC*, 2013.

[8] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[9] Hilde Kuehne, Juergen Gall, and Thomas Serre. An end-to-end generative framework for video segmentation and recognition. *WACV*, 2016.

[10] Colin Lea, Michael D. Flynn, Rene Vidal, Austin Reiter, and Gregory D. Hager. Temporal convolutional networks for action segmentation and detection. *CVPR*, 2017.

[11] Colin Lea, Austin Reiter, Rene Vidal, and Gregory D. Hager. Segmental spatiotemporal cnns for fine-grained action segmentation. *ECCV*, 2016.

[12] Behrooz Mahasseni, Sheng Chen, Alan Fern, and Sinisa Todorovic. Detecting the moment of snap in real-world football videos. *IAAI*, 2013.

[13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR*, 2013.

[14] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. *ICML*, 2013.

[15] Zhongang Qi and Fuxin Li. Embedding deep networks into visual explanations. *ICLR*, in press.

[16] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ArXiv e-prints*, September 2014.

[17] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. *CVPR*, 2016.

[18] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. *ICCV*, 2015.

[19] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *IJCV*, 2015.

[20] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *ECCV*, 2014.

[21] Jianming Zhang, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. Top-down neural attention by excitation backprop. *ECCV*, 2016.