Deep Learning Course Project: Automatic Football Highlights Editor

Zitian Tang
2019012398
Institute for Interdisciplinary Information Sciences
Tsinghua University
tzt19@mails.tsinghua.edu.cn

Zimeng Song
2019012379
Institute for Interdisciplinary Information Sciences
Tsinghua University
songzm19@mails.tsinghua.edu.cn

Fangxuan Liu
2019012387
Institute for Interdisciplinary Information Sciences
Tsinghua University
liufx19@mails.tsinghua.edu.cn

1 Introduction

Football is one of the most popular and exciting sports around the world. Football game highlights, a collection of the most exciting and vital parts in games, are an efficient way for audiences to recap the games. As usual, an entire football game video is more than 90 minutes, while the highlights are highly concentrated and less than 10 minutes, containing about ten scenes. In convention, football highlights are manually edited, requiring editors to go through the whole video and carefully select the most significant scenes.

Many deep learning techniques were developed to replace manual jobs in the past decade, making it promising for AI to edit football highlights automatically. Besides, multimodal learning is a field that fusing data in different modalities to help the model achieve better performance. In this course project, we develop a deep learning model to generate football highlights from the full game video automatically, using both video and audio information.

In specific, we design an automatic highlight editor model with two stages. The first stage is called scene classifier, used to detect essential scenes from a full game video. The second stage is called precise scene editor, which can precisely find the start point and endpoint for each scene detected by the scene classifier. Section 4 introduce our model in details. Our codes and demos can be found in our GitHub.[1]

We also do some ablation studies to see how our method performs comparing to single modal ones. Described in Section 6.1 and 6.2, our multimodal method outperforms the one only using audio. However, it is outperformed by only using video, which is a common but complex problem in the multimodal field.

[1] https://github.com/AutomaticHighlights/AutomaticHighlights

Preprint. Under review.

Moreover, an image dataset called SEV Dataset proposed in [2] is used for pretraining in our method. We construct an image dataset from our collected data by ourselves to study whether we can build up our model without SEV Dataset. However, described in Section 6.3, we finally find that using SEV Dataset can help our model achieve better performance.

2 Related works

Most methods for automatic football highlights are based on either video or audio. [3] uses 369 game videos and corresponding highlights to develop a deep learning method based on "excitement recognition" through speech analysis, containing two stages, one for scene classification and the other for producing continuous highlight results. However, different from how we use audio information, this work uses Google Speech-to-Text API to convert the audio into the text to obtain good performance. [5] develops an automatic highlights model based on the video, with the live text of football matches as auxiliary information. This work uses some tricks like tracking the time and scoreboard and applying OCR on it.

[1] proposes a multimodal framework to summarize football events according to user's selection, but not for automatical highlights. They use a BiLSTM to calculate the importance of video clips and classify them into several event classes. Finally, the scenes fitting the user's selection are assembled.

[2] uses deep learning to detect events in football games based on image classification. To achieve this, a football event image classification dataset called SEV Dataset is introduced. This dataset gives us a choice for pretraining when developing our method.

3 Data collection and preprocessing

We collect more than 400 full game videos, and their corresponding manually generated highlights from CCTV.com, covering six different football leagues: CFA Super League, UEFA Champions League, Serie A, Ligue 1, AFC Champions League, and Libertadores. To ensure the high quality of our data, we only collect the videos of games from August 2019 to March 2021.

In order to get both positive and negative samples, we use some conventional methods to precisely find the locations of highlights in the original full game videos (as illustrated in Figure 1). Specifically, we calculate the ℓ_2-distance and Normalized Cross-Correlation (NCC) between frames in highlights and original videos, then set a threshold and match them by some rules to ensure the order of frames are not changed after matching.

Figure 1: Illustration of video preprocessing. We match the frames between highlights and original videos to find the locations of highlights in each game.

We finally successfully match 364 games. Serving for the consequent training, we divide these games into the training set and validation set. The statistics of our dataset are shown in Table 1.

4 Model framework

In this section, we first introduce the overall pipeline of our model, which is composed of two stages: scene classifier and precise scene editor. After that, we introduce the frameworks of the two stages, respectively.

Table 1: Statistics of dataset

	Training set	Validation set	Total
Number of games	331	33	364
Number of highlight scenes	3907	424	4331

4.1 Overall pipeline

To generate highlights from a full game video, our pipeline contains two stages:

- Stage 1, scene classifier: use a classifier to predict how likely each scene in the game is expected to be chosen into highlights.
- Stage 2, precise scene editor: use a model to precisely find the start point and endpoint for each scene detected by Stage 1.

The major reason we induce the second stage rather than using only a classifier to generate highlights is that we find scene classifier can correctly classify scenes but is not sensitive to the most vital part in a scene (e.g., it may notice the celebration and replay to recognize goals rather than directly recognizing goals).

4.2 Stage 1: scene classifier

The purpose of the scene classifier is to give each scene (around 20s∼60s) confidence about how likely this scene should be chosen as a highlight scene.

Due to the limitation of computation resources, we first extract features of video and audio using pre-trained feature extractors and then combine them to train a multimodal classifier. The specific framework of the scene classifier is shown in Figure 2. We use Log-Mel Spectrogram as an audio feature, which is widely used in audio recognition, and use a pre-trained model as an image feature extractor to extract video features. These two types of features are concatenated together as scene features, and a sequence model (in practice, we use BiLSTM) gives a highlight probability based on the features.

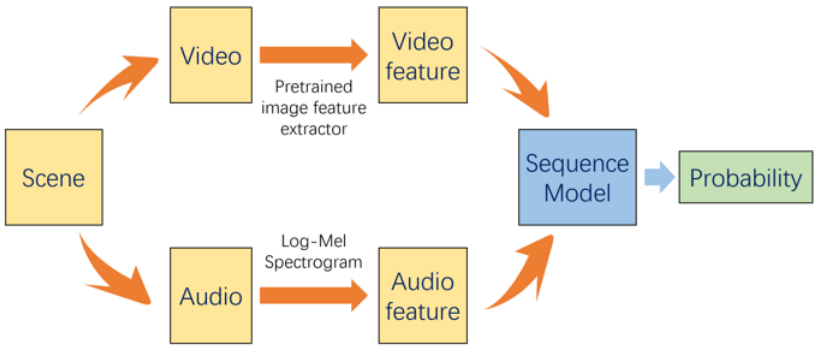


Figure 2: Framework of scene classifier. In practice, we use a BiLSTM as the sequence model.

As for the image feature extractor, we train a model on the football event image classification dataset (SEV Dataset) proposed in [2], and use intermediate outputs as features. Besides, we also study whether we can build our model without this dataset, so we construct a binary image classification dataset by sampling images from game videos and highlights and use it to pre-train. In the following, this dataset constructed by ourselves is called Highlight Dataset. More details about this pretraining process are described in Appendix A, and the comparison between results using these two datasets is discussed in Section 6.3.

Notice that we have divided each game video into two types of scenes, highlight scenes and non-highlight scenes, by preprocessing. When training the scene classifier, we use the highlight scenes as positive samples and non-highlight ones as negative samples. Since non-highlight scenes are much longer than highlight ones, we crop the non-highlight scenes randomly with the same length distribution as highlight scenes, which also avoids classifying according to video length. Besides, to augment the dataset, we randomly crop a part of $4/5$ length from each sample during training.

To detect highlight scenes from an entire game video through a scene classifier, we use a sliding window with a length of 30s and stride of 1s and predict the highlight probability for each window. After that, for each frame, we average the probabilities of the windows covering it and obtain a probability curve, like Figure 3. Then the scenes with a probability above a threshold are detected as highlight scenes.

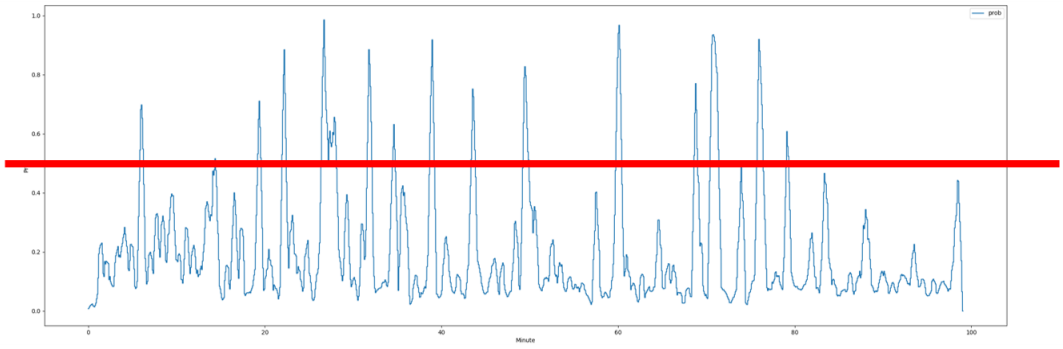


Figure 3: An example of probability curve generated by scene classifier.

4.3 Stage 2: precise scene editor

The purpose of a precise scene editor is to precisely find where each detected scene in Stage 1 starts and ends.

Different from the scene classifier, the precise scene editor should give each frame a score in a scene and find an interval with high scores. We view this task as a segmentation task over temporal dimension where the highlight scenes are foregrounds, and non-highlight scenes are backgrounds, and classifying each frame into the two classes according to its context.

For training data, we prepare the following dataset: for each piece of highlight scenes (t_l, t_r), we extract features of $(t_l - t, t_r + T - t)$, where t is uniformly sampled from $[0, T]$ and T is set to 60s to avoid the model from memorizing t if it is fixed. The features in the original highlight are marked as foreground, and the remaining features are marked as backgrounds.

There are two families of models to handle this task: we can use models for serial data, e.g., LSTM. We can also modify the 2-dimensional segmentation models into a 1-dimensional version, a hand-crafted 1-dimensional U-net, for example. The empirical results prefer the previous family, which can be explained by the property of the training target that the foreground is always continuous, and serial model can easily capture this property for its arbitrarily long reception field. After some trials, we choose BiLSTM as our final model as it can directly capture the property of the target and its good performance in similar tasks.

After deriving this precise scene editor, we use it to revise the scenes detected in Stage 1. Specifically, for each highlight scene, we extend 30s at its beginning and the end, then use the precise scene editor to predict a probability curve for this scene. We clip the longest two intervals with a probability larger than 0.5 into the final highlights.

5 Results

In our final model, we choose the image feature extractor trained on SEV Dataset rather than that on Highlight Dataset.

The training results of the first stage (scene classifier) are shown in Table 2. Our model achieves an accuracy of 0.84 on the validation set. Notice that our recall is a little lower compared to precision and specificity, which happens in [3] as well. Model, in this case, may miss some essential parts in the game but provides highlights with less meaningless scenes.

Table 2: Stage 1 (scene classifier) training results

Metric	Training set	Validation set
Accuracy	0.913	0.843
Precision	0.900	0.875
Recall	0.932	0.807
Specificity	0.893	0.880

The training results of the second stage (precise scene editor) is shown in Table 3. Our model achieves accuracy around 0.83 on validation set in this task.

Table 3: Stage 2 (precise scene editor) training results

Metric	Training set	Validation set
Accuracy	0.946	0.826

To see the actual generation results, we test our model on several recent football games. The generated highlights can be found in our GitHub.[2] In the following, we discuss how our model performs on these games.

Most of the goals in these games, which are the most vital parts in football games, are successfully captured by our model. However, we find the reason for this success is that our model is sensitive to replays. We used to try only using a scene classifier to generate highlights, but in many cases, the replay of a goal is selected into the highlights while the goal itself is not. Fortunately, our precise scene editor helps us deal with many of these cases. The precise scene editor learns to clip a goal from the start point of the attack.

However, our model is not sensitive to the red card, which is supposed to be included in the highlights. Besides, our model sometimes collects scenes that are not so vital while some other essential scenes are missed. The major reason is that the probabilities given by our scene classifier cannot represent the importance precisely, though they are much higher than those of unimportant scenes. Decreasing the threshold is a way to improve the recall, but the highlights will be much longer if we do so.

Our model's sensitivity to replays also causes another issue: when it is replaying a goal that happened a long time before, our model cannot distinguish it is happening currently and selects it into the highlights.

6 Ablation studies

We do ablation studies to check whether our multimodal model performs better than only using audio or video, which are discussed in Section 6.1 and 6.2, respectively. Besides, in Section 6.3, we study how our model performs when replacing the SEV dataset with Highlight Dataset during pretraining the image feature extractor.

Convenient for the following discussions, we show the training results of the models used in ablation studies of Stage 1 on validation set in Table 4, and those of Stage 2 in Table 5.

6.1 Use audio only

As shown in Table 4 and 5, our final model can achieve much higher accuracy than only using audio, indicating that video can supply much complementary information for this task. We also try using this audio model to generate game highlights. We find that the model can still successfully capture

[2] `https://github.com/AutomaticHighlights/AutomaticHighlights`

Table 4: Stage 1 training results of different methods on validation set

Method	Accuracy	Precision	Recall	Specificity
Final model	0.843	0.875	0.807	0.880
Only use audio	0.713	0.750	0.656	0.773
Only use video	**0.863**	0.864	**0.868**	0.858
Pretrain using Highlight Dataset	0.833	0.835	0.837	0.829
Use two feature extractors	0.854	**0.900**	0.802	**0.907**

Table 5: Stage 2 training results of different methods on validation set

Method	Accuracy
Final model	**0.826**
Only use audio	0.747
Pretrain using Highlight Dataset	0.813
Use two feature extractors	0.820

the most vital scenes in a game, such as goals, but it may include more meaningless scenes than our final model into the highlights.

6.2 Use video only

From Table 4, we can see only using video can achieve higher validation accuracy with higher recall. This looks like a failure of our multimodal training. In fact, this phenomenon that multimodal training derives worse validation performance than single modal is a problem faced for years in the multimodal field. There is no perfect solution to this problem yet. In this case, multimodal provides more information to the model so that the model memorizes the training set rapidly and achieves bad generalization. We find our model is also stuck in this problem since the multimodal one achieves lower validation accuracy and overfits the training set more quickly than the other one.

We also compare the differences in highlights generated by our final model and the video model. Comparing to the final one, the video model is more sensitive to replay scenes, where the camera's position is different from the usual. It infers whether a scene is significant according to whether there is a long-time replay. Although our final model sometimes relies on this information as well, it can also capture scenes by other evidence. This is a reason why we choose the multimodal one as our final model.

6.3 Use Highlight Dataset for pretraining

After the two image classifiers on SEV Dataset and Highlight Dataset are well-trained, we first use them to generate highlights to see whether they are sensitive to the same features. First, set a fixed length (30 seconds) for each scene. Then calculate the sum of classified probabilities for each frame in a scene. Finally, select 20 scenes (about 10 minutes in total) among a game (about 90 minutes). In this way, we extract the highlight for a soccer game with our image classifiers.

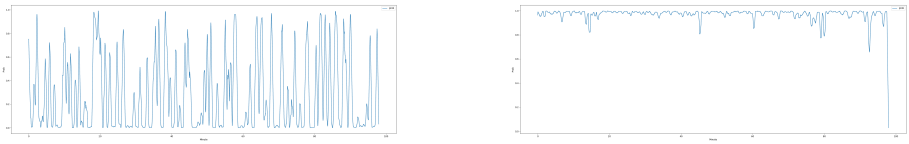
We find that two different classified models trained on two different datasets prefer scenes distinctly (Fig 4). The model trained on SEV Dataset gives us more scenes around soccer inside the soccer field (Fig 4(a)). On the other hand, the model trained on our Highlight Dataset gives us more scenes about human figures (Fig 4(b)), including players, coaches, and fans. This indicates that two models trained on different dataset capture different features of images.

(a) SEV Dataset's highlights (b) Highlight Dataset's highlights

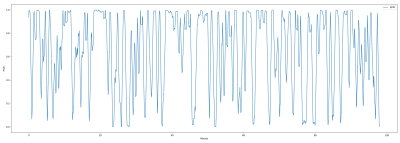
Figure 4: Two models trained on two datasets capture distinct features

To see how Highlight Dataset affects our model performance, we tried three methods in Stage 1 and 2: using feature extractor from SEV Dataset (final model), using feature extractor from Highlight Dataset, and using both two extractors.

Training results of Stage 1 are shown in Table 4. Using a feature extractor from Highlight Dataset can only achieve worse performance than that from SEV Dataset, but using both two extractors can achieve higher accuracy. Nonetheless, when generating highlights using models with Highlight Dataset features, the confidence given by the model is not reliable. Figure 5 shows the probability curves generated by these three different models for the same game. Models involved with the Highlight Dataset feature extractor give high confidence to much more scenes, making them incredible. However, surprisingly, even in this case, the most significant scenes still enjoy the highest confidence.



(a) SEV Dataset feature extractor (b) Highlight Dataset feature extractor



(c) Using both feature extractors

Figure 5: Probability curves generated by models with different image feature extractors in Stage 1 for the same game.

Training results of Stage 2 are shown in Table 5. In this task, feature extractor from Highlight Dataset again performs worse than that from SEV Dataset, but using both two feature extractors doesn't help improve the performance. We think this is again because of the multimodal training issue mentioned in Section 6.2.

7 Discussion

In this project, we develop a 2-stage deep learning model using both video and audio information to automatically edit highlights from a full football game video. This model first uses a scene classifier to detect vital scenes in a game, then uses a precise scene editor to precisely find the start and endpoint of each scene. Our model performs well in some of the recent football games but also meets some issues.

The biggest drawback of our method is that the generated highlights are very long (around 10 minutes). The major reason is it doesn't have the ability to select the most vital scenes among all the detected scenes in Stage 1. For example, when making highlights for a game, if there are many goals in this game, other less brilliant attacks should take less place in the highlights. However, when our model gives a confidence score to a scene, it only looks at local information rather than the overall game. We come up with a method to deal with this problem but have no time to try: after Stage 2, extract a feature for each scene, then use a Transformer to select the most important ones among these dozens of scenes.

Another regrettable thing is that our work is stuck in a hard problem in the multimodal field. Although our multimodal model can generate highlights with better visual effects, it fails to achieve better evaluation statistics on the validation set comparing to the model only using video. We believe a more effective multimodal fusion method can bring the stronger capability to our model.

Besides, the COVID-19 pandemic has a bad effect on our game video data. Since most of our collected games are during the COVID-19 pandemic period, there are only a few or even no spectators in these games. Notice that spectators can have a great influence on both video and audio information; our model is likely to perform a little worse in games with many spectators.

Acknowledgments and Disclosure of Funding

We would like to thank Prof. Hang Zhao for the discussion about the phenomenon that multimodal training derives worse training results than only using video.

References

[1] T. Haruyama, S. Takahashi, T. Ogawa, and M. Haseyama. User-selectable event summarization in unedited raw soccer video via multimodal bidirectional lstm. *Multimedia Tools and Applications*, 9(1):42–53, 2021.

[2] A. Karimi, R. Toosi, and M. A. Akhaee. Soccer event detection using deep learning. *CoRR*, abs/2102.04331, 2021.

[3] V. Scotti, L. Sbattella, and R. Tedesco. Sferanet: Automatic generation of football highlights. In *6th International Conference on Computer Science, Engineering and Information Technology (CSEIT-2019)*, volume 9, page 101, 2019.

[4] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *CoRR*, abs/1905.11946, 2019.

[5] K. Tang, Y. Bao, Z. Zhao, L. Zhu, Y. Lin, and Y. Peng. Autohighlight : Automatic highlights detection and segmentation in soccer matches. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4619–4624, 2018.

A Details about pretraining image feature extractor

A.1 Method

We want to train a classification model that can recognize soccer events or judge whether an image is in a highlight first. When this model achieves high accuracy in the image classification task, it may capture image characteristics in intermediate layers. Here we directly use the values of the last intermediate layer to be the feature of the image.

Refering to a previous soccer event classification task of [2], we use EfficientNetB0 [4]. Unlike [2] using a Variational Autoencoder module to detect the soccer event first and use fine-grain classification module in the end, we remove this two parts because we do not need an utimate classification model here. We focus on feature extracting. So we directly use EfficientNetB0 to be trained on the dataset, then use the last intermediate layer (shown in Fig 6)as the image feature.

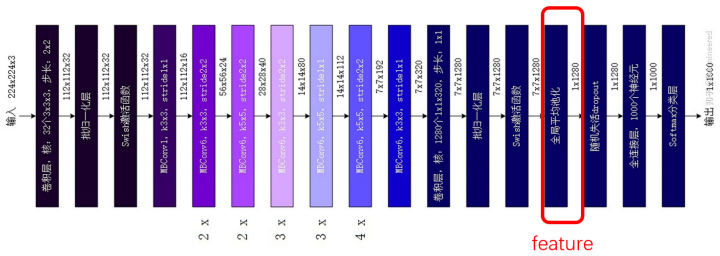
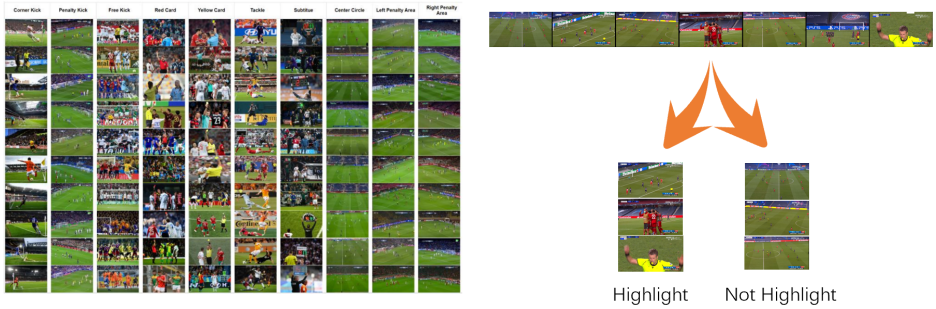


Figure 6: Feature extractor structure

A.2 Dataset

For training the classification model, we use two classification datasets (Fig 7) about soccer. Training on these two different datasets gives us different image features. The first dataset is the soccer event (SEV) dataset (Fig 7(a)) introduced by [2]. This dataset contains nine categories of soccer events, including corner kick, penalty kick, free kick, red card, yellow card, tackle, to substitute, center circle, left penalty area, and right penalty area. The second dataset is our highlight dataset (Fig 7(b)). According to our labeled highlight data, we simply split the video and labeled each image as 'in highlight' or 'not in highlight', two categories.



(a) Soccer Event Dataset (b) Our Highlight Dataset

Figure 7: Two Different Datasets for Training the Classification Model

A.3 Evaluation

We evaluate two classification models that are trained on two different datasets respectively (Table 6). Accuracy on SEV Dataset is much higher than that on our Highlight Dataset because the former dataset is more elaborate.

Table 6: Accuracy on two datasets

Dataset	Training set	Validation set
SEV Dataset	0.99	0.89
Highlight Dataset	0.84	0.76

We also check the classified result with confidence on real-world soccer game videos. We attach the largest probability of classification result on each frame. Some examples are shown in Fig 8. Most of the events in nine categories are classified correctly, but some events like goals cannot be recognized since there are no corresponding labels during training. Besides, a variety of bright colors can confuse the model. For example, the country flags can be mistakenly classified as red or yellow cards, but that indeed makes sense.

Figure 8: Classification results on SEV Dataset