Spotlight: A Smart Video Highlight Generator
Stanford University CS231N Final Project Report

Jun-Ting (Tim) Hsieh
junting@stanford.edu

Chengshu (Eric) Li
chengshu@stanford.edu

Wendi Liu
wendiliu@stanford.edu

*Kuo-Hao Zeng
khzeng@cs.stanford.edu

Abstract

During the past few years, we have witnessed an unprecedented growth of user-generated videos (UGV). Being able to efficiently and effectively process unedited videos has become increasingly important. In this project, we aim to improve video understanding by extracting the fundamental and most interesting parts out of videos.

The model we propose consists of two layers: 1) feature extraction, where we turn windows of frames into feature vectors and 2) highlight classification, where we categorize feature vectors as highlights or non-highlights of the video. For each layer, we experiment with various mainstream computer vision algorithms. We obtain a dataset of 3,700 UGVs including manually-labeled highlight intervals. Our best-performing model achieves an F1 score of 0.433. Our model also performs well in a real-world setting with human users, who give positive feedback on our model output.

1. Introduction

With the increasing popularity of content-sharing websites and live-streaming platforms such as YouTube, Vimeo and Livestream, video has become the most common form of Internet traffic. We are constantly immersed in videos from all kinds of sources. However, the amount of information we receive everyday makes it impossible for anyone to process all of them. Despite the amazing progress we have made in classifying and describing images, we cannot yet achieve the same with videos. Motivated to help people understand videos in the most efficient and effective manner, we want to extract the fundamental components of videos. In this project, we propose a deep learning model that automatically identifies the important highlights of videos.

The focus of this project is on user-generated videos (UGV), videos that are created and shared by users through on-line video-sharing communities. Therefore, the videos we work with are not domain-specific, but are very diverse, featuring a wide range of topics and styles. This makes our project particularly interesting but also challenging.

In order to identify highlights in a video, we use a window-based classification model. First, we split a video into windows of 64 frames. Then, for each window, we use our model to classify whether it is a highlight or not. The reason why we use a window-based model, as oppose to a frame-based model (classifying frame by frame), is that it gives us access to both the temporal information of consecutive frames within the window and the relationship between windows in a video.

The main goal of this project is to experiment with different algorithms for the substructures in our model and compare their performances on this specific task of highlight classification. First, we will explore different ways to encode the windows of a video. Then, we will compare models that capture temporal information with models that do not. Finally, we will look at different regularization methods and see whether they improve the performance.

2. Related Work

2.1. Feature Extraction

The first part of our model generates feature vectors using features extracted from videos. One popular feature-learning algorithm is Convolutional 3D, or C3D, developed by Tran el al. [1]. C3D learns spatio-temporal features using deep 3-dimensional convolutional networks trained on video datasets such as the UCF-101 dataset. Traditional image-based deep-learning frameworks are also able to achieve this task. Improving on previous Google Inception models, Szegedy el al. [2] propose the Inception-v3 model, which is both computationally efficient and utilizes very few parameters. It Iandola el al. [3] use smaller DNN architectures that achieves AlexNet-level [14] accuracy on traditional image classification tasks. Both these models are pre-trained on ImageNet. However, they do not capture

*This co-author contributed the dataset for this project.

temporal information in the video.

2.2. Highlight Classification

Many early highlight detection works focus on specific categories of sport videos, including basketball, baseball and cricket. [4, 5, 6, 7] These models mostly rely on low-level visual features. Recently, a few methods have been proposed to find highlights in generic personal videos. Sun et al. [8] train a model to identify domain-specific highlights through harvesting user preference. There have also been a few attempts to use fully unsupervised learning. Zhao and Xing [9] propose a quasi-real time method to generate short video summaries. Yang et al. [10] propose a recurrent auto-encoder to extract video highlights. Our model adopts a supervised learning approach. Previously, Zeng et al. [11] build a model that generates title for UGVs. For the highlight detection part of their model, they train a bidirectional RNN highlight detector. This framework is able to capture temporal information, which is very important to videos. It accomplishes outstanding results for subsequent tasks in their model. Yeung el al. [13] introduce a reinforcement learning algorithm for efficient video processing that could speed up training time by 5 times. Ma el al. [15] develop a viewer attention model specifically for video summarization task. We plan to incorporate these newer methods in our model as future works.

3. Dataset

We use the same dataset consisting of user-generated videos as used by Zeng el al. [11] in the paper "Title Generation for User Generated Videos", in which they train a highlight detector as an intermediate step for their video captioning task. In the dataset, each video comes with a hand-labeled interval of the start and end frame of the highlight in this video. For example, a video with 1000 frames may have a highlight interval of (200, 800).

We pre-process the dataset by filtering out unusually long or short videos and videos with corrupted data. Our final dataset contains 3687 videos (around 3900000 frames). Overall, the median video length is around 800 frames, or 30 seconds, and the median highlight length is around 100 frames, or 3 seconds.

Next, we break each video into windows of 64 frames, as shown in Figure 1. A window is considered a highlight if more than 75% of the frames in this window is within the highlight interval, i.e., more than 48 frames of the window is within the highlight interval. In the end, we have around 180000 windows in total.

Ideally, we would like to store all the windows of a video as a single unit and classify the windows all at once. However, due to the limitation of the length of an RNN structure, we decide to split long videos into shorter clips. We define a clip to be 32 consecutive windows. Since the number of windows of a video usually is not divisible by 32, for a clip that has less than 32 windows, we zero-pad it to 32 windows and use a boolean mask to indicate valid length.

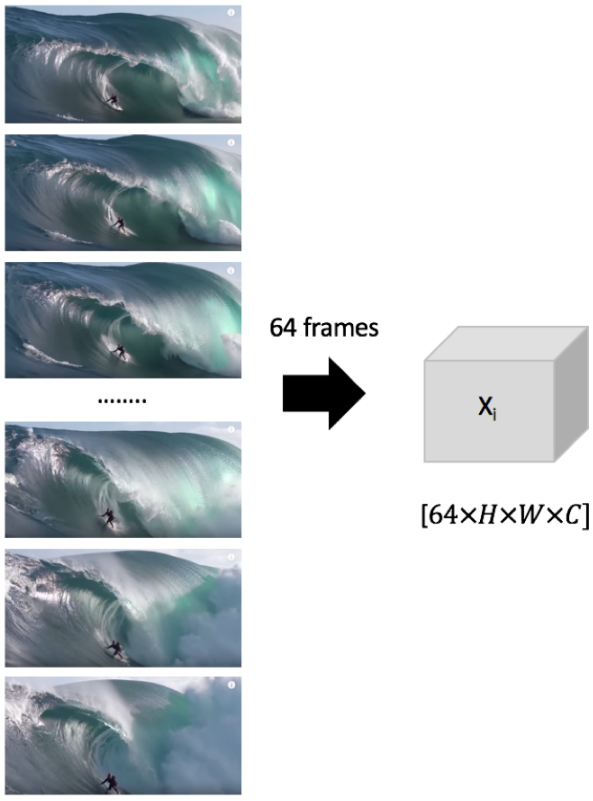

Figure 1. Each window is a tensor that represents 64 consecutive frames

In total, we generate 5616 clips, which are further split into train/val/test sets of 4491/562/563 clips.

In summary, our dataset consists of 5616 clips. Each clip consists of 32 consecutive windows from a single video, and each window is 64 consecutive frames of that video. Each window is also labeled 0 or 1, indicating whether it's a highlight or not.

4. Methods

The model we are proposing is built using a two-layer structure2 3. The first layer transforms each window into a one-dimensional vector. The second layer takes this vector and classifies it as highlight or non-highlight. In this paper, we will refer to the first layer as the base model, the one-dimensional vector as a feature vector, and the second layer as the top model.

Figure 2 and figure 3 illustrate the two-layer structure of our model. Each X_i is a window that consists of 64 frames. Our model outputs two class scores for highlight and non-highlight for each window.

Our entire model is basically a transfer learning model,

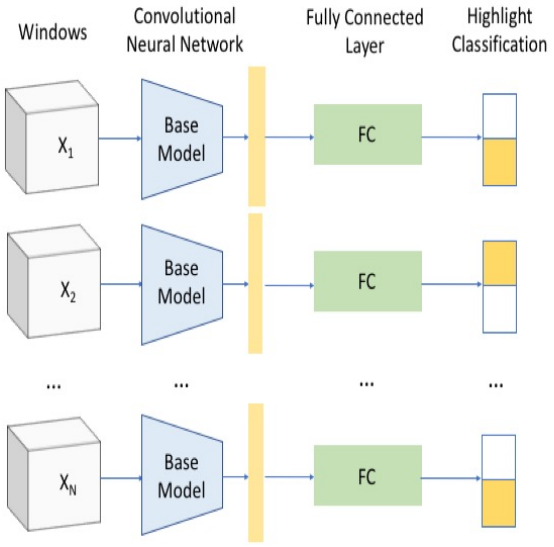

Figure 2. Two-Layer Model with Fully Connected Top Layer

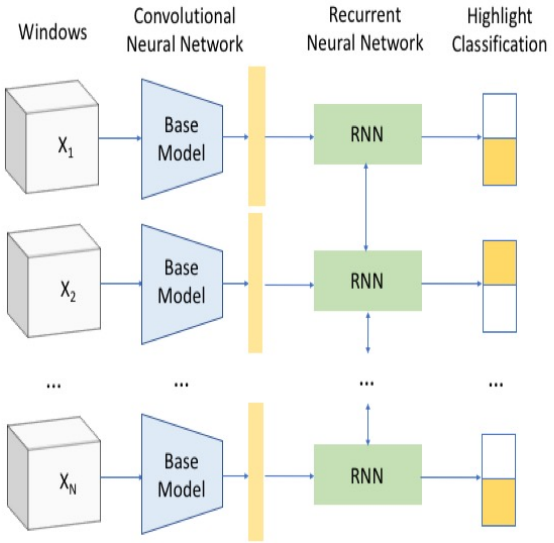

Figure 3. Two-Layer Model with Recurrent Neural Network Top Layer

where we take the last layer of a pre-trained model as our feature vector, and add layers on top for a binary classification. During training, we do not fine-tune the base model.

4.1. Base model

Our base model transforms each window into a one-dimensional vector. We experiment with 4 different base models: Inception, C3D, C3D-normalized, and SqueezeNet.

4.1.1 Inception-v3

Since a window consists of 64 frames, we evenly sample 8 frames and feed them into a pre-trained Inception-v3 model.[2][1] Then, we extract the final layer of Inception, which is a 2048-dimensional vector. Thus, we have eight 2048-dimensional vectors for each window. Finally, we average these eight vectors to form our feature vector. Since we average the vectors, we lose the temporal information of the frames in a window.

4.1.2 C3D

A pre-trained C3D model [1] takes in 16 frames at a time, and it's final layer (fc7) is a 4096-dimensional vector. Thus, we divide each window into 4 segments of 16 frames, feed them into the C3D model [16], which gives us back 4 vectors. Then, we average these 4 vectors to get our feature vector. This model is able to capture the temporal information of frames in a window.

4.1.3 C3D-normalized

This model is similar to the C3D we describe in 4.1.2, but is slightly more complicated. We split each window into 16 frame long segments with an 8-frame overlap between two consecutive segments. Thus, we have 7 segments for each window. Then, we extract the fc6 activations from the C3D model and average them to form a 4069-dimensional vector. Finally, we normalize it using L2 normalization to get our final feature vector.

This method is suggested by Tran et al. [1] for using C3D as a feature extractor for video analysis tasks. Similar to the C3D model, this model is able to capture the temporal information of frames in a window.

4.1.4 SqueezeNet

Iandola et al. [3] have shown that SqueezeNet has AlexNet-level accuracy with much smaller size and fewer parameters. Similar to the Inception-v3 model, we evenly sample 8 frames from each window and extract the last layer vector of SqueezeNet, which is 512-dimensional. We average them to form our feature vector. Similar to the Inception-v3 model, we lose the temporal information of the frames in a window.

4.2. Top model

The inputs of the top model are the vectors generated by the base model. The top model outputs two class scores (non-highlight, highlight) for each window, which are subsequently used to calculate loss and make predictions. We experiment with 2 top models: fully-connected and bidirectional LSTM. We will refer to these models as FC and LSTM hereafter.

4.2.1 FC

We use 3 fully-connected layers with output sizes of 2048, 2048 and 2, respectively. The first two layers are followed by ReLU activations. Even though our dataset consists of clips with consecutive windows, this model treats all windows independently, which means we do not capture the relationship between windows.

4.2.2 LSTM

Since our clips are 32 windows long, the base model will output 32 feature vectors for each video clip. We feed these into a bidirectional LSTM of length 32. Then, we add 2 fully-connected layers on top of the hidden states, with the first followed by ReLU activation. The hidden size of the LSTM is 512, and the output sizes of the linear layers are 256 and 2. Unlike the previous model, the LSTM captures the relationship between windows.

4.3. Training

The top model generates two class scores for each window. We try two different types of loss function: cross-entropy loss and hinge loss. For parameter updates, we use the Adam optimizer to update the top model variables, and we go through hyperparameter search to fine-tune the learning rate. For regularization, we try adding dropout and batch normalization between fully-connected layers in our top model.

4.4. Evaluation

To evaluate our model performance, we look at the F1 score, which is the harmonic mean of the precision and recall. More specifically, precision is the fraction of correct predictions among the positive predictions (windows predicted to be highlight), and recall is the fraction of ground-truth positive (highlight) windows that are successfully retrieved. Since both are meaningful and relevant to this task, the F1 score is a good metric to evaluate our model.

On the other hand, we do not look at accuracy, which is the fraction of correct predictions among all predictions. This is because most windows in our dataset are non-highlights, and a model that always predicts windows as non-highlights will have a good accuracy. We are more interested in the highlight windows, so the F1 score is a better metric.

During each training session, we calculate the F1 score on the validation set after every epoch. We save the checkpoint which achieves the highest F1 score on the validation set, and then use this checkpoint to calculate the F1 score on the test set.

5. Experiments and Analysis

As discussed above, we select 4 algorithms for our base model and 2 algorithms for our top model. For each pair of base-top model pair, we run experiments with the following settings:

- Adam optimizer with 6 different learning rates: $10^{-3.5}$, 10^{-4}, $10^{-4.5}$, 10^{-5}, $10^{-5.5}$, 10^{-6}.
- 2 different loss functions: cross-entropy loss, hinge loss.
- With or without the regularizations including:
  - Dropout with rate 0.5
  - Batch normalization between fully-connected layers (only for the FC top model).

5.1. Overall

We have 8 combinations of the base and top model. We run all the experiments and record the F1 score on the test set (as described in section 4.4). Table 1 shows the best F1 scores and the hyperparameters that produce the result for each combination.

Our best model is Inception-v3, LSTM, which achieves an F1 score of 0.433.

5.2. Base model

For base model, Inception-v3 is the best among the 4 base models we try. Originally, we expect that C3D and C3D-normalized would perform the best, since C3D is able to capture the temporal information of frames within a window, i.e., the trajectory of the video. However, there are several other factors that must be taken into account. Most importantly, Inception-v3 is pre-trained on ImageNet, while C3D is pre-trained on the UCF-101 dataset. These are two totally different datasets, and it turns out that the pre-trained feature extractor for ImageNet performs slightly better for our specific video highlighting task.

5.3. Top model

With the same base model, LSTM performs better than FC in most cases. This is the same as we expected, since it captures the relationship between the feature vectors generated from the base model. In other words, our bidirectional LSTM is able to learn from the relationship between consecutive windows, which corresponds to the large-scale motion of the video.

5.4. Learning rate

We found that the learning rates 10^{-4} and $10^{-4.5}$ (for Adam optimizer) generally give the best results. Figure 4 shows the validation F1 scores after each epoch of training

Base, top model	F1 score	Learning rate	Loss function	Regularization
Inception-v3, FC	0.370	$10^{-4.5}$	Cross-entropy	0.5 dropout
Inception-v3, LSTM	**0.433**	$10^{-4.5}$	Cross-entropy	None
C3D, FC	0.325	10^{-4}	Cross-entropy	None
C3D, LSTM	0.370	$10^{-3.5}$	Cross-entropy	None
C3D-normalized, FC	0.393	$10^{-3.5}$	Cross-entropy	Batch normalization
C3D-normalized, LSTM	0.362	10^{-4}	Cross-entropy	None
SqueezeNet, FC	0.338	10^{-5}	Cross-entropy	None
SqueezeNet, LSTM	0.371	10^{-4}	Cross-entropy	None

Table 1. Best test F1 scores overall and the hyperparameters that produce the result

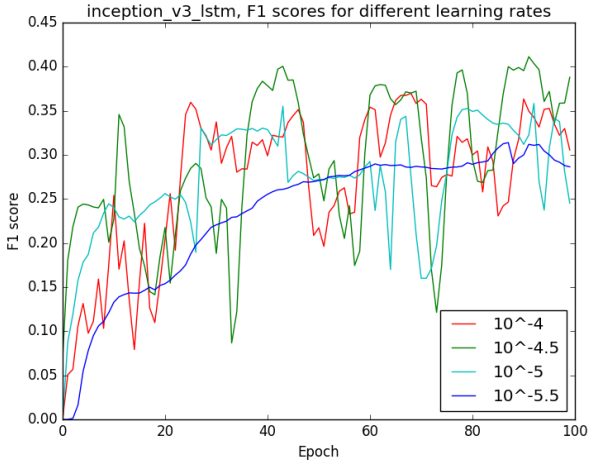

Figure 4. Inception-v3, LSTM. Validation F1 scores after every epoch during training, using different learning rates. Here we left out the curve for learning rates $10^{-3.5}$ and 10^{-6} since they have even worse F1 scores.

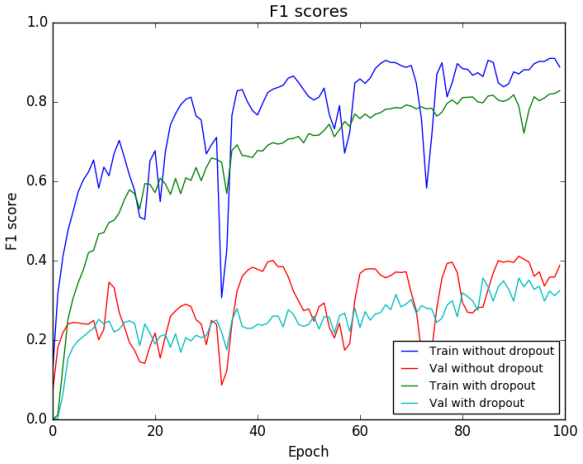

Figure 5. Inception-v3, LSTM. Training F1 score v.s. validation F1 score with dropout (dropout rate is 0.5) and without dropout.

using different learning rates. From the graph we can see that the curve for learning rate $10^{-4.5}$, which is our best model, has the highest validation F1 score.

As shown in Figure 4, the validation F1 scores for the higher learning rates fluctuate a lot, while the curve for learning rate $10^{-5.5}$ is relatively smooth. This indicates that our loss landscape requires a smaller learning rate. Thus, we tried learning rate decay, but we did not see an improvement. This will be discussed in section 5.7.

5.5. Loss function

Using cross-entropy loss performs better than using hinge loss for all combinations. Hinge loss also seems to take longer to train and harder to converge.

5.6. Regularization

As shown in Figure 5, our model's training F1 score is much higher than the validation or test F1 score. We tried to use dropout with a dropout rate of 0.5 to alleviate overfitting. However, the result is less than satisfactory. While dropout helps stabilize the training process, it does not help close the gap very much. Moreover, it even lowers the validation and test F1 scores.

We also tried adding batch normalization between fully-connected layers. We only added batch normalization between the fully-connected layers of the FC top model. Similarly, we did not see an improvement for our model.

The results of our experiments with different regularizations are shown in Table 2. In most cases, adding dropout or batch normalization does not increase or even decreases our F1 scores.

5.7. Learning rate decay

We tried to decay the learning rate every 20 epochs by 0.5 in the logarithmic scale. Our hope is that by decaying learning rate, we can bring down the loss even further after it plateaus. However, although our F1 score curve is slightly smoother, we achieved worse results after deploying learning rate decay.

Figure 6 shows the train and validation F1 scores with and without learning rate decay for Inception-v3, LSTM. We see that the curves with decay are smoother, but the

Base, top model	No regularization	Dropout	Batch Normalization
Inception-v3, FC	0.355	0.370	0.361
Inception-v3, LSTM	0.433	0.386	N/A
C3D, FC	0.325	0.282	0.323
C3D, LSTM	0.370	0.336	N/A
C3D-normalized, FC	0.386	0.356	0.393
C3D-normalized, LSTM	0.362	0.347	N/A
SqueezeNet, FC	0.338	0.334	0.329
SqueezeNet, LSTM	0.371	0.319	N/A

Table 2. Best test F1 scores for different regularization methods. We did not add batch normalization in our LSTM top model.

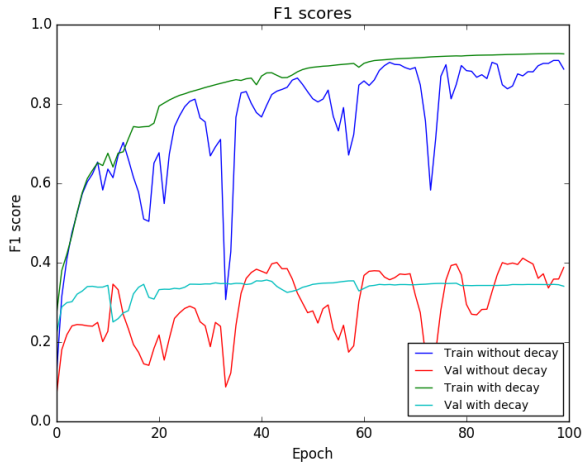

Figure 6. Inception-v3, LSTM. Training F1 score v.s. validation F1 score with and without learning rate decay

model without decay reaches a higher validation F1 score.

Our hypothesis is that even though the F1 scores fluctuate a lot for higher learning rates, the fluctuations might randomly push the model to a good validation F1 score, which then result in a good test F1 score.

6. Conclusion and Future Work

Through extensive hyperparameter search, we achieve decent performance on our video highlight classification task using the Inception-v3 and LSTM model. Considering the diversity of topics and styles of user-generated videos and the subjectivity of video highlights, we believe our model has achieved satisfactory results.

For our base model, even though C3D is expected to perform better because it is able to capture the temporal information between frames, we find that Inception-v3 actually performs better. We suggest that this is because the pre-trained weights of C3D are trained on the UCF-101 dataset and its feature vectors either do not generalize to our video highlighting task or not as well as the feature vectors from Inception-v3.

For our top model, we find that a bidirectional LSTM performs better than fully-connected layers on the same base model. This shows that understanding the relationship between consecutive windows in a video does boost the performance, as is expected.

In addition, we try several regularization methods to reduce over-fitting. However, it turns out that adding dropout or batch normalization does not improve our performance. Moreover, while decaying learning rate during training reduces the F1 score fluctuations, it lowers test F1 scores as a consequence.

In conclusion, the best model we build uses Inception-v3 for our base model and LSTM for our top model. It achieves 0.433 test F1 score. We think that this project has a lot of actual use cases. Therefore we also build a mobile application on top of it, which we bring to actual human users. We get a lot of positive feedback on the model outputs, which shows that it does have potential to help people better understand videos.

We have quite a few ideas in mind in terms of future work. First, we can use state-of-the-art traditional CV feature extractor such as dense trajectory developed by Wang el al. [11] and then build fully-connected or LSTM layers on top of it. We also want to experiment with attention model both across image pixel dimension and across time dimension. Inspired by Yeung el al [13], we would love to incorporate reinforcement learning and non-uniform and hence efficient video processing in our model as well.

References

[1] Tran D., Bourdev L., Fergus R., Torresani L. and Paluri M. "Learning spatiotemporal features with 3D convolutional networks". ICCV, 2015.

[2] Szegedy, C., Vanhoucke V., Ioffe S., Shlens J. and Wojna, Z. "Rethinking the Inception Architecture for Computer Vision". arXiv:1512.00567v3, 2015.

[3] Iandola, F. N., Moskewicz, M. W., Ashraf, K., Han, S., Dally, W. J. and Keutzer, K. "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡ 1mb model size". arXiv:1602.07360, 2016.

[4] Nepal, S., Srinivasan, U. and Reynolds, G. "Automatic detection of goal segments in basketball videos". ACM Multimedia, 2001.

[5] Kolekar, M. and Sengupta, S. "Event-importance based customized and automatic cricket highlight generation". ICME, 2006.

[6] Hanjalic, A. "Adaptive extraction of highlights from a sport video based on excitement modeling". EEE Transactions on Multimedia, 2005.

[7] Tang, H., Kwatra, V., Sargin, M. and Gargi, U. "Detecting highlights in sports videos: Cricket as a test case". ICME, 2011.

[8] Sun, M., Farhadi, A. and Seitz, S. "Ranking domain-specific highlights by analyzing edited videos". ECCV, 2014.

[9] Zhao, B. and Xing, E.P. "Quasi real-time summarization for consumer videos". CVPR, 2014.

[10] Yang, H., Wang, B., Lin, S., Wipf, D., Guo, M. and Guo, B. "Unsupervised extraction of video highlights via robust recurrent auto-encoders". ICCV, 2015.

[11] Zeng, K. H., Chen, T. H., Niebles, J. C. and Sun, M. "Title Generation for User Generated Videos". ECCV, 2016.

[12] Wang, H., Klaser, A., Schmid, C. and Liu, C. L. "Action Recognition by Dense Trajectories". IEEE, 2011.

[13] Yeung, S., Russakovsky, O., Mori, G., Li, F. "End-to-end Learning of Action Detection from Frame Glimpses in Videos". CVPR, 2016.

[14] Krizhevsky A., Sutskever I., Hinton G. E. "ImageNet Classification with Deep Convolutional Neural Networks". NIPS, 2012.

[15] Ma, Y.F., Lu, L., Zhang H.J. and Li, M. "A User Attention Model for Video Summarization". ACN, 2002.

[16] Our C3D and C3D normalized model is built upon GitHub repository C3d-tensorflow. https://github.com/hx173149/C3D-tensorflow

[17] Our Inception model is built upon Tensorflow Slim. https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/slim

[18] Our SqueezeNet model is built upon assignment 3. http://cs231n.github.io/assignments2017/assignment3/