Finding Interesting Moments in Videos: Audio-visual and Unsupervised Video Highlight Detection

by

Taivanbat Badamdorj

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Signal and Image Processing

Department of Electrical and Computer Engineering
University of Alberta

Abstract

Video is the dominant medium through which we consume information. Due to the sheer volume of video content, it is of great interest in the research community to develop automated ways to make this content manageable. In video highlight detection, our goal is to automatically find interesting moments in videos. This makes it much easier to browse, search, categorize, and edit large amounts of video data. In this thesis, we develop two new state-of-the-art methods for video highlight detection. We first explore the use of audio for highlight detection. This is largely unexplored territory despite audio providing useful cues for finding interesting moments. Then we develop a new unsupervised method that alleviates the need for manual annotations that tell us exactly which parts of a video are interesting. Although our main focus is on video highlight detection, our models are general and we believe that they are applicable to other tasks such as action recognition, and video classification.

Preface

This thesis is an original work by Taivanbat Badamdorj. The work in Chapter 3 has previously appeared in the following publication by the author:

- T. Badamdorj, M. Rochan, Y. Wang, and L. Cheng, "Joint visual and audio learning for video highlight detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8127–8137.

To my dad, mom, and brother.

Acknowledgments

I would like to thank my supervisor, Prof. Li Cheng for his continuous support, wonderful discussions, and openness to new ideas. Prof. Cheng has an unwavering optimism and calmness when it comes to research. I would also like to thank Prof. Yang Wang and Dr. Mrigank Rochan for their guidance and incisive questions that would help me to uncover the core of the questions that I was trying to answer. Thanks to Prof. Cheng, Prof. Wang, and Dr. Rochan, I was never short of ideas. In addition, I would like to thank my labmates for their productive discussions and friendship. Special thanks to Chuan Guo, Shihao Zou, Mahdiar Nekoui, Javad Khaghani, Ji Yang, and Hoang Nguyen.

I also want to thank all of my friends who made Edmonton feel like home: Aditya Jayaprakash, who is the best roommate I have ever had. Not only do we share a similar sense of humor, I could always count on him to sharpen my thoughts. He's one of the smartest people I know. On top of that, his cat, Misty, brought immense joy to both our lives. Through Aditya, I met Archit Sakhadeo, whom I still owe a meal for when I visit Ontario (his obsession with Persian cuisine will never be forgotten). He is one of the most methodical thinkers that I know of. I loved having laughter-filled meals together with Aditya and Archit.

I'm grateful for all the friends that I made in Edmonton and all the friends who stayed in touch with me from elsewhere. In no particular order and without any guarantees that this list is complete, I am thankful for Sara Sibilo, Koyee Wong, James Suh, Alexis Alford, Kira Eberts, Janisse Breitkreuz, Pauleanne Codilla, Jacob Sarkadi, Olivia Menard, Nana Amoh, Onon Bayasgalan, Galit Greenberg, Ayman Iraqi, and Shaul Hafner.

Finally, and with the greatest emphasis, I would like to thank my family: my dad, mom, and brother. Though separated by distance, they were still near in my heart. I want to thank my parents for all the sacrifices they've made for me to be here. I want to thank my brother, Tsetsenbat Badamdorj, for being a pillar of support and being level-headed when perhaps I could not be. It's been amazing to see your growth and maturity and I am very proud of you. My family's support has helped me all the way from being a mischievous six-year old entering the first grade, to now, wrapping up my masters.

Table of Contents

1 Introduction **1**

1.1 Thesis Contributions 3

1.2 Thesis Outline 3

2 Related Work **4**

2.1 Video Highlight Detection 4

2.1.1 Background 4

2.1.2 Prior efforts 5

2.2 Video Summarization 6

2.3 Multi-modal Learning 6

2.4 Visual Sentinel 7

2.5 Contrastive Learning 7

3 Audio-visual Learning for Video Highlight Detection **9**

3.1 Our Approach 11

3.1.1 Unimodal Self-Attention 12

3.1.2 Bimodal Attention 14

3.1.3 Noise Sentinel 15

3.1.4 Classifier 16

3.1.5 Model Learning 17

3.2 Experiments 18

3.2.1 Datasets and Setup 18

3.2.2 Baselines 20
3.2.3 Highlight Detection Results 21
3.2.4 Ablation Studies 24
3.3 Summary 26

4 Contrastive Learning for Unsupervised Video Highlight Detection 27
4.1 Approach 29
4.1.1 Generic Attention Layer 30
4.1.2 Video Encoder 33
4.1.3 Contrastive Learning 34
4.2 Experiments 36
4.2.1 Datasets and Setup 36
4.2.2 Unsupervised Highlight Detection 38
4.2.3 Why Does It Work? 40
4.2.4 Effect of dropout on performance 42
4.2.5 Qualitative Results 43
4.3 Summary 44

5 Conclusion 45

Bibliography 47

List of Tables

3.1 Audio-visual highlight detection results (mAP) on the YouTube dataset . . 19

3.2 Audio-visual highlight detection results (top-5 mAP) on the TVSum dataset 20

3.3 Audio-visual highlight detection results (mAP) on the VTW dataset 21

3.4 Audio-visual model: comparison to baselines 22

3.5 Audio-visual model: missing modality at test time 24

3.6 Cross-dataset testing results 25

3.7 Ablation study on disentangled attention 26

3.8 Ablation study on noise sentinel 26

4.1 Comparison to unsupervised and weakly supervised methods on the YouTube dataset (mAP) 37

4.2 Comparison to unsupervised and weakly supervised methods on TVSum dataset (mAP at top-5) that don't use external data 38

4.3 Highlight detection results (mAP) on the VTW dataset 38

4.4 Comparison to unsupervised and weakly supervised methods on TVSum dataset (mAP at top-5) that use external data 39

4.5 Comparison to supervised methods on the YouTube dataset (mAP) 40

4.6 Cluster quality metrics for non-highlight and highlight clips 41

4.7 Effect of dropout on performance for contrastive learning 42

List of Figures

2.1 Example of highlight detection problem 5

3.1 Audio classification probabilities over time 10
3.2 Audio-visual highlight detection framework 11
3.3 Bimodal attention layer 14
3.4 Attention map for video with and without noise sentinel mechanism 17
3.5 Qualitative results for audio-visual model 23

4.1 Highlight clips vs non-highlight clips to cluster videos 28
4.2 Contrastive learning framework 30
4.3 Unsupervised model inference at test time 31
4.4 Generic attention layer 32
4.5 Video encoder architecture 33
4.6 Qualitative results for unsupervised model 43

Chapter 1

Introduction

We have witnessed an explosion of online video content in recent years due to the rapid adoption of video based social networks like YouTube, Instagram, and TikTok. This has led to increasing demands for video highlight detection, which aims to automatically detect interesting moments (called "highlights") within a video. Highlight detection is important due to its broad range of downstream applications including video summarization, recommendation, editing, and browsing. In consequence, there has been significant progress in the field in recent years [1–11].

In video highlight detection, we divide a video into clips of some fixed length, and output a score for each clip that indicates the likelihood that it is a highlight. It is fundamentally a ranking problem where we want to output scores such that highlight clips have a higher score than non-highlight clips for the video.

Video highlight detection has a broad range of applications. For example, we can imagine that on video sharing websites such as YouTube, we can improve video browsing by detecting interesting parts of long videos so that users can save time and skip ahead. Another example is video surveillance since it is a time consuming process for humans to sift through security footage. An automated method that finds the moments of interest in the security footage may save hours of time. It may also be useful in video editing. For example, generating a trailer from a full length movie is a long process that may be sped up by a highlight detection method that finds the most interesting moments within the movie, and

generates a first draft for a trailer. A user could then edit the draft according to their taste [11].

We can also use the detected highlights to generate a short summary of the video. Although closely related to video highlight detection, the video summarization task has a slightly different aim. In video summarization, the task is to output a set of segments that will represent the entire video, rather than just a score for every clip in the video. It aims to distill the video into a shorter version of itself that still retains the original meaning of the video. Thus for the video summarization task, we may also want to take into account factors such as redundancy (we do not want two clips that are similar to both be a part of the summary), and representativeness (we want to select clips that represent the main idea). Besides saving time for the user, it also saves time for other video based applications such as video recommendation and action recognition. Fundamentally, we remove the meaningless parts of the video, thus providing a "cleaner" signal for downstream video-based applications.

Despite recent progress, there are still several areas in which video highlight detection is lacking. Firstly, the vast majority of existing approaches do not take into account the effect of audio. Video is a multi-modal medium, so it makes sense that interesting moments also depend on the audio. We want to be able to find what is interesting by hearing and by seeing, as these two modalities often work together to determine what is interesting to us. Secondly, most existing approaches depend on manually annotated training data. This means that for every video, a human annotator must first watch the entire video, then manually tag the interesting clips. As one can imagine, this is a very time-consuming and expensive process. Thus, we are interested in developing an unsupervised method that does away with manual supervision altogether.

1.1 Thesis Contributions

In this thesis, we develop two methods for video highlight detection. We make the following contributions:

- We propose a new audio-visual highlight detection model in Chapter 3. Unlike prior methods that only use visual features, we develop a multi-modal framework that captures the interaction between the audio and visual features. In this chapter, we also present a simple formulation that allows us to deal with the case where the audio or visual features are noisy. We achieve state-of-the-art results on the highlight detection task.

- We present a novel *unsupervised* video highlight detection framework in Chapter 4 that learns to pick highlight clips in order to perform video clustering through contrastive learning. Unlike prior weakly-supervised and unsupervised methods, we do not collect a large external dataset. We outperform prior works in weakly-supervised and unsupervised methods in video highlight detection.

1.2 Thesis Outline

In Chapter 2, we discuss related work. In Chapter 3, we present a novel method for detecting highlights that jointly uses audio and video. In Chapter 4, we present our unsupervised highlight detection framework. In Chapter 5 we conclude this thesis and discuss possible future avenues of research.

Chapter 2

Related Work

In this chapter, we review some related works on highlight detection as well as tasks related to highlight detection. Then we review work from other tasks that inspire some of our methods.

2.1 Video Highlight Detection

2.1.1 Background

The goal of video highlight detection is to find interesting moments in videos. In Fig. 2.1, we show an example where the highlight moments are outlined in green. In this parkour video, we want to find the parts of the video where people are actually doing parkour, instead of just uninteresting background parts where there is nothing of interest happening. In practice, existing works in highlight detection split a video into clips of a fixed size (*e.g.* 100 frames), obtain vector features for the frames within the clip using a pre-trained neural network such as a C3D action recognition model [12], and take the average of the vector features within the clip. For each clip, we thus obtain a single vector representation. Then our goal is to output a scalar value called a highlight score for the clips in the video such that highlight clips have a higher score than non-highlight clips.

Figure 2.1: Example of highlight detection problem. In this parkour video, we want to find the parts where people are doing parkour.

2.1.2 Prior efforts

Early efforts in video highlight detection mainly deal with sports videos [13–15]. Later works were proposed to deal with videos from social media [1], as well as first-person videos [16].

Some works formulate highlight detection as a classification problem [9], while it is also popular to formulate highlight detection as a ranking problem [1, 2, 5–7, 11], where a ranking network is trained to rank highlight clips higher than non-highlight clips.

Recently, weakly-supervised and unsupervised highlight detection methods have been proposed [8, 10, 17–21], where the training label is only available at the entire video level. The work of [8] takes advantage of the fact that clips from shorter videos are more likely to be highlights to train a ranking network. MN [10], on the other hand, proposes a multiple instance ranking framework that learns to rank clips from a given category higher than clips from other categories.

MN [10] is the only other work that proposes an audio-visual framework. Their network operates independently for each clip, and uses a variant of simple concatenation to produce a fused audio-visual feature. In contrast, our model in Chapter 3 can utilize the context within the entire video through the self-attention and bimodal attention layers. In addition, while their work assumes that audio is useful only as a complementary feature to the visual features, we make no such assumption, and allow each modality to modulate the other.

Many weakly supervised methods also rely on the collection of large datasets to train their models [8, 10, 17], in contrast to our unsupervised highlight detection method in Chapter 4. We compare our method to prior works in Chapter 4.

2.2 Video Summarization

Video summarization is a closely related task aimed at producing a compact and cohesive summary of a given video. Early works in video summarization are predominantly unsupervised [19, 20, 22–30], and as such, many rely on heuristics such as diversity and representativeness to obtain a summary video.

Weakly supervised methods [17, 21–23, 27, 31, 32] have also been developed to utilize video-level information. Benefiting from the massive user tagging of online videos, research efforts in supervised learning [28, 33–39] are also progressing rapidly.

Our work is influenced by the attention-based model of [39] that operates on sequences of clips as potential highlights.

2.3 Multi-modal Learning

Recent progress has been made in multi-modal learning in various fields such as action recognition [40], and speech recognition [41, 42]. It has been observed [43] that simple fusion strategies often fail. We also observe in our experiments in Chapter 3 that late and early fusion strategies are suboptimal, and propose the use of a bimodal attention layer which captures the interaction between modalities.

Our work is also related to audio-visual speech recognition [41, 42]. The work of [41] uses an attention network but does not model the interplay between audio and visual channels, instead the output of the previous decoding layer is used as the query for both modalities. The work of [42] considers a one-way relation where the audio can be used to query the visual features, but not vice-versa. In contrast, our model in Chapter 3 is symmetric, and allows the visual features to query the audio features.

2.4 Visual Sentinel

The visual sentinel is a concept introduced in image caption generation by [44]. The task of image caption generation is to generate a textual description for a given image. The key idea is that caption generation models do not always need to attend to the image; words such as“and”,“of”, and “from” have no visual counterpart.

Therefore, the visual sentinel [44] is proposed as a “look-away” mechanism. When generating words without visual grounding, their model learns to attend to the visual sentinel instead of the image.

This inspires us to formulate a *noise sentinel* in our context to automatically “look away” from a modality if it is noisy. As we shall see in the experiments in Chapter 3, this greatly improves the empirical performance of our approach.

2.5 Contrastive Learning

The idea of contrastive learning is to make representations of a sample agree under small transformations [45]. In the case of images, we can transform an image by applying augmentations such as random cropping, random jittering, or random flips. Then we train a network to output a representation that is similar to the same image with a different set of transformations applied to it. In essence, we learn to form a cluster for each sample where each member is a transformed version of the sample. A neural network trained on such a task would then output a representation of an image that is useful for downstream tasks such as image classification.

In other words, we aim to learn effective representations by learning to pull semantically close neighbors and pushing apart non-neighbors [46]. While the transformations to be applied are clear in the case of images, it is unclear exactly how to apply transformations to discrete modalities such as text. Recent work has suggested using dropout [47] within the model as a form of data augmentation [48]: we can get different representations for

the same input by using different random dropout masks. The final task would then be the same as the case for images: pull the two representations of the same sample together, while pushing the representation farther away from the representations of other samples in the training dataset.

We use this idea in Chapter 4 for unsupervised video highlight detection. Our video encoder picks the most important clips that help us to differentiate between embeddings of different videos.

Chapter 3

Audio-visual Learning for Video Highlight Detection

In this chapter, we explore the use of audio in video highlight detection. Although the audio component of the video provides important cues for highlight detection, the majority of existing efforts focus almost exclusively on the visual component. In this chapter, we argue that both audio and visual components of a video should be modeled jointly to retrieve its best moments.

Let's take a look at an example: Fig. 3.1, shows a video of a car crashing during a race. We may identify the interesting part of the video by seeing the crash. On the other hand, hearing people stop cheering and start talking could also be an indicator that the moment is interesting. While we can process the visual and audio information separately, our insight is that they also interact with each other. We can imagine that if we were in the crowd depicted in Fig. 3.1, we would look to see what had happened if everybody suddenly stopped cheering. It can also work the other way around: what we hear can reinforce our beliefs that the given moment is interesting. These observations motivate us to propose an approach that jointly learns from visual and audio information to detect highlights in videos. It consists of two different attention mechanisms: a unimodal self-attention mechanism that models the relationships between moments belonging to the same modality; and a bimodal attention mechanism that models the interaction between the two modalities.

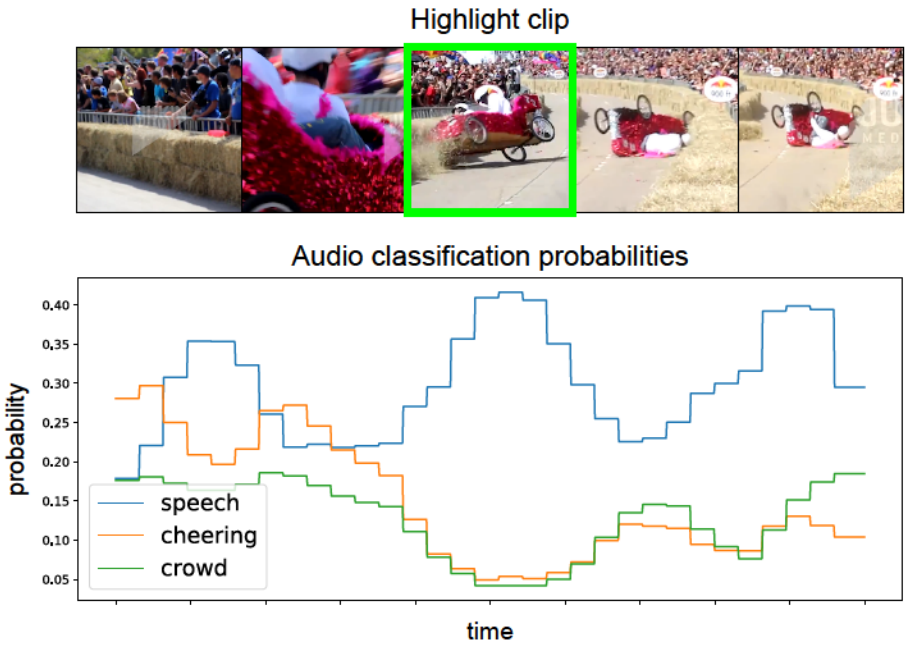

Figure 3.1: Audio-visual highlight detection: audio can be informative about which part of the video is a highlight. In this video, a car crashes during a race in front of a large crowd. The highlight (in green) is the crash, and we also show the top three audio class probabilities from a pre-trained audio classification network [49]. The class probabilities (speech, cheering, and crowd) show that the cheering and crowd noise dies out during the crash, while talking increases. Intuitively, we understand that if people stop cheering suddenly, something must have happened. In this work, we learn to utilize such audio cues.

Furthermore, we impart the ability to ignore a modality on our model. Intuitively, if we hear something interesting, but do not see anything interesting when we look, we would like to be able to ignore what we hear. In addition, if we do not hear anything of interest, it is also not worth looking. Inspired by the visual sentinel [44], we introduce a noise sentinel in the bimodal attention mechanism that allows our model to "look-away" from a modality by attending to the noise sentinel instead.

The proposed attention mechanisms and the noise sentinel are novel and effective as empirically indicated in the ablation studies. We demonstrate our model's superior performance on three well-known benchmarks, where our approach significantly outperforms the state-of-the-art methods.

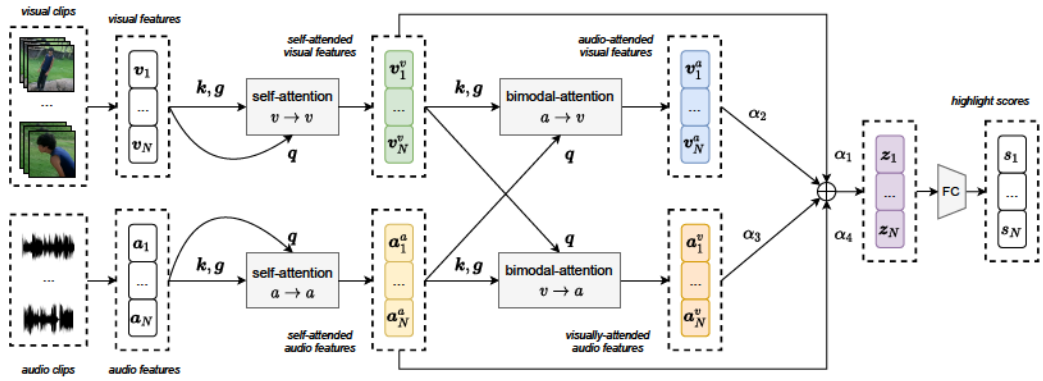

Figure 3.2: An overview of our framework. An input video is evenly split into clips of a fixed length, producing N total clips. The visual and audio modalities of the i-th clip are then represented as vectors, $\boldsymbol{v}_i$ and $\boldsymbol{a}_i$, respectively. For each modality, the relationship between clips are tackled separately in the self-attention layer; this leads to the self-attended features $\boldsymbol{v}_i^v$ for visual, and $\boldsymbol{a}_i^a$ for audio. It is then fed into the bimodal attention layers to model the cross-modality relationships, and produce the bimodally attended features $\boldsymbol{v}_i^a$ (audio attended visual features) and $\boldsymbol{a}_i^v$ (visually attended audio features). A more detailed inspection of the bimodal attention layer is shown in Fig. 3.3. Finally, the self-attended and bimodally attended features come together with a learned weighted sum, to produce the feature $\boldsymbol{z}_i$, which is processed to output the final clip highlight score s_i for the i-th clip.

3.1 Our Approach

We show the overall architecture of our framework in Fig. 3.2. We first split a video V into clips of a fixed length (*e.g.*100 frames), resulting in N clips. We characterize the i-th clip by two vectors, $\boldsymbol{v}_i$ for the visual features, and $\boldsymbol{a}_i$ for the audio features, where $i \in 1, 2, ..., N$. These visual and audio features are extracted using pre-trained visual [12, 50] and audio feature extractors [49]. Moreover, as explained in Fig. 4.5, we capture the interactions between clips using two different attention mechanisms, namely self-attention and bimodal attention. For each modality, the self-attention layer captures interactions among clips of the same modality. The output of both modalities are then fed to the two input branches of the bimodal attention layers which capture the interplay between the audio and visual clips. These attended features are subsequently fused and fed to a classifier which produces a highlight score s_i for the i-th clip, which indicates the likelihood that the clip is a highlight.

3.1.1 Unimodal Self-Attention

A given clip by itself is often inadequate to determine whether it is a highlight; rather it is beneficial to consider the relationship of this clip with other clips in the video. The self-attention mechanism [51] and its variants have been shown to be effective in modeling such dependencies. Our self-attention mechanism is an adaptation of the disentangled attention block [52] to our context. Without loss of generality, we describe in what follows our self-attention mechanism for the visual features, $\{\boldsymbol{v}_i\}_{i=1}^{N}$.

Let us denote the visual self-attention relationship with $v \rightarrow v$ (we attend from visual to visual features). We denote by $W_{q,v\rightarrow v}$, $W_{k,v\rightarrow v}$, and $W_{m,v\rightarrow v}$ linear projection matrices, and define the following quantities:

$$\boldsymbol{q}_{v\rightarrow v}(\boldsymbol{v_i}) = \hat{\boldsymbol{q}}_{v\rightarrow v}(\boldsymbol{v_i}) - \frac{1}{N}\sum_{l=1}^{N}\hat{\boldsymbol{q}}_{v\rightarrow v}(\boldsymbol{v_l}), \tag{3.1}$$

$$\boldsymbol{k}_{v\rightarrow v}(\boldsymbol{v_j}) = \hat{\boldsymbol{k}}_{v\rightarrow v}(\boldsymbol{v_j}) - \frac{1}{N}\sum_{l=1}^{N}\hat{\boldsymbol{k}}_{v\rightarrow v}(\boldsymbol{v_l}), \tag{3.2}$$

Here $\boldsymbol{q}_{v\rightarrow v}(\boldsymbol{v_i})$ and $\boldsymbol{k}_{v\rightarrow v}(\boldsymbol{v_j})$ are called the query and the key in self-attention. $\hat{\boldsymbol{q}}_{v\rightarrow v}(\boldsymbol{v_i})$ and $\hat{\boldsymbol{k}}_{v\rightarrow v}(\boldsymbol{v_j})$ represent linear projections. We subtract the mean of these linear projections from each quantity in Eq. (3.1) and Eq. (3.2).

$$\hat{\boldsymbol{q}}_{v\rightarrow v}(\boldsymbol{v_i}) = W_{q,v\rightarrow v}\boldsymbol{v}_i, \tag{3.3}$$

$$\hat{\boldsymbol{k}}_{v\rightarrow v}(\boldsymbol{v_j}) = W_{k,v\rightarrow v}\boldsymbol{v}_j. \tag{3.4}$$

We also define the term m_j, which is a linear projection of each key into a scalar real. m_j is called the unary term.

$$m_j = W_{m,v\rightarrow v}\boldsymbol{v}_j. \tag{3.5}$$

Now, the visual self-attention score $\omega_{v\rightarrow v}(\boldsymbol{v}_i, \boldsymbol{v}_j)$ between two clips $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ is defined

by

$$\omega_{v\to v}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \text{softmax}(c\, \boldsymbol{q}_{v\to v}(\boldsymbol{v}_i)^\intercal \boldsymbol{k}_{v\to v}(\boldsymbol{v}_j)) + \text{softmax}(m_j), \tag{3.6}$$

where the softmax is over the key indices j (over the attended sequence). Following [51], the constant c is used to normalize the dot product and to improve gradient flow through the softmax operator.

Intuitively, the attention score $\omega_{v\to v}(\boldsymbol{v}_i, \boldsymbol{v}_j)$ captures the amount of dependency of $\boldsymbol{v}_i$ on $\boldsymbol{v}_j$ relative to other clips in the video. As suggested by [52], the pairwise term $\boldsymbol{q}_{v\to v}(\boldsymbol{v}_i)^\intercal \boldsymbol{k}_{v\to v}(\boldsymbol{v}_j)$ captures the relationship between clips, while the unary term m_j represents the saliency of every clip within the video. This allows our module to learn a representation that takes into account the relationship between clips as well as the overall importance of each clip in the entire video.

We define the linear projection matrix, $W_{g,v\to v}$. The attended features are then produced by applying attention scores to re-weight the input features,

$$g_{v\to v}(\boldsymbol{v}_j) = W_{g,v\to v}\boldsymbol{v}_j, \tag{3.7}$$

$$\boldsymbol{v}_i^v = \sum_{j=1}^{N} \omega_{v\to v}(\boldsymbol{v}_i, \boldsymbol{v}_j) g_{v\to v}(\boldsymbol{v}_j) + \boldsymbol{v}_i, \tag{3.8}$$

where $g_{v\to v}(\boldsymbol{v}_j)$ is called the value embedding of $\boldsymbol{v}_j$. The attended feature vector $\boldsymbol{v}_i^v$ is the combination of the original feature $\boldsymbol{v}_i$ and the sum of other clip features weighted by their corresponding attention scores. Here we adopt the convention of using superscript to denote the modality that is used as a query. For example, $\boldsymbol{v}_i^v$ refers to the visually-attended visual features of the i-th clip.

The self-attended features of the audio modality ($a \to a$) can be similarly defined as:

$$\boldsymbol{a}_i^a = \sum_{j=1}^{N} \omega_{a\to a}(\boldsymbol{a}_i, \boldsymbol{a}_j) g_{a\to a}(\boldsymbol{a}_j) + \boldsymbol{a}_i. \tag{3.9}$$

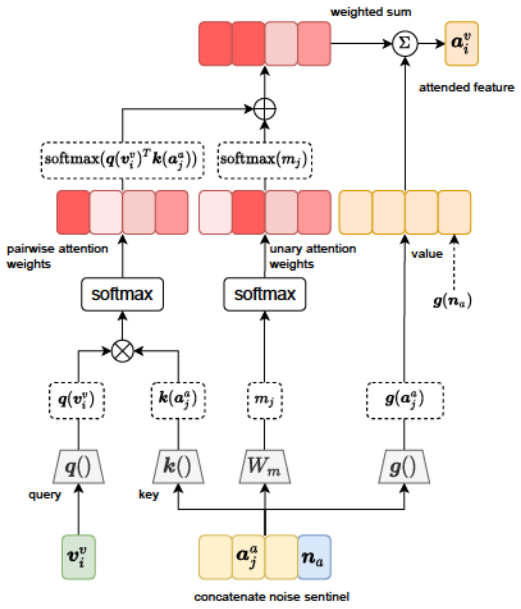

Figure 3.3: Detailed illustration of the bimodal attention layer. It aims to capture the relationship between the audio and visual modalities. The figure showcases attending to audio using visual features ($v \rightarrow a$). The noise sentinel is a part of the bimodal attention layer, formed by concatenating along the time axis to the modality attended to. The bimodal attention is then computed according to Eq.(3.19).

3.1.2 Bimodal Attention

The self-attention only captures the clip interactions within one modality. To capture the interactions across modalities, we extend the self-attention mechanism and devise a bimodal attention, shown in Fig. 3.3. In our bimodal attention, we compute the query from one modality, and the key and value from the other modality. This is presented in the following case of using visual features to attend to the audio, i.e. $v \rightarrow a$, as:

$$\boldsymbol{a}_i^v = \sum_{j=1}^{N} \omega_{v\rightarrow a}(\boldsymbol{v}_i, \boldsymbol{a}_j) g_{v\rightarrow a}(\boldsymbol{a}_j) + \boldsymbol{a}_i^a. \tag{3.10}$$

Here $\omega_{v\rightarrow a}(\boldsymbol{v}_i, \boldsymbol{a}_j)$ and $g_{v\rightarrow a}(\boldsymbol{a}_j)$ are similarly defined w.r.t. Eqs. (3.1)–(3.6). The bimodal attention mechanism allows the information from two different modalities to influence each other.

3.1.3 Noise Sentinel

Given the features from the self-attention and bimodal attention, a simple strategy is to directly fuse these features (e.g. via concatenation) for final prediction. Empirically, we have found that this strategy does not give the best performance. This is possibly due to the noise in the features. For example, if the audio features are noisy, the visually attended audio features from the bimodal attention layer are not reliable. In addition, using the noisy audio to query the visual features would also result in unreliable visual features. Intuitively, we would like our model to have the capacity to ignore a certain modality when it is noisy. Inpired by the visual sentinel [44] used in image captioning, we present a novel noise sentinel mechanism for this purpose.

The noise sentinel is a parameter within the bimodal attention layer. It has the same channel-dimension as the attended features within the attention layer, and is initialized to zero. It is then concatenated to each of the self-attended sequences as follows:

$$[\boldsymbol{a}_1^a, \boldsymbol{a}_2^a, \ldots, \boldsymbol{a}_N^a, \boldsymbol{n}_a], \tag{3.11}$$

$$[\boldsymbol{v}_1^v, \boldsymbol{v}_2^v, \ldots, \boldsymbol{v}_N^v, \boldsymbol{n}_v], \tag{3.12}$$

where $\boldsymbol{n}_a$ and $\boldsymbol{n}_v$ represent the noise sentinel parameters. We use the self-attended visual features $\boldsymbol{v}_i^v$ as queries to attend to the sequence in Eq. (3.11). Symmetrically, we use the the self-attended audio features $\boldsymbol{a}_i^a$ to attend to the sequence in Eq. (3.12). Note we do not use the noise embedding as a query. In what follows, we derive the visually-attended audio features $\boldsymbol{a}_i^v$. The audio-attended visual features $\boldsymbol{v}_i^a$ can be obtained by following-suit.

Let us denote the key for the audio noise embedding $\boldsymbol{n}_a$ as $\boldsymbol{k}_{\text{noise}}$, and the unary term for the noise embedding as m_{noise}. We define a few more variables for notational convenience.

$$\boldsymbol{q}_i = \boldsymbol{q}_{v\to a}(\boldsymbol{v}_i^v), \tag{3.13}$$

$$\boldsymbol{k}_j = \boldsymbol{k}_{v\to a}(\boldsymbol{a}_j^a),\ \boldsymbol{k}_{\text{noise}} = \boldsymbol{k}_{v\to a}(\boldsymbol{n}_a), \tag{3.14}$$

$$m_j = W_{m,v\to a}\boldsymbol{a}_j^a,\ m_{\text{noise}} = W_{m,v\to a}\boldsymbol{n}_a, \tag{3.15}$$

$$g(\boldsymbol{a}_j^a) = W_{g,v\to a}\boldsymbol{a}_j^a,\ g(\boldsymbol{n}_a) = W_{g,v\to a}\boldsymbol{n}_a. \tag{3.16}$$

We are ready to define the attention scores. The attention score from the visual to audio clips is

$$\begin{aligned}\omega_{v\to a}(\boldsymbol{v}_i^v, \boldsymbol{a}_j^a) =& \frac{\exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_j)}{\exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_{\text{noise}}) + \sum_{l=1}^{N} \exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_l)} \\ &+ \frac{\exp(m_j)}{\exp(m_{\text{noise}}) + \sum_{l=1}^{N} \exp(m_l)};\end{aligned} \tag{3.17}$$

and the attention score between the visual clip and audio noise embedding is

$$\begin{aligned}\omega_{v\to a}(\boldsymbol{v}_i^v, \boldsymbol{n}_a) =& \frac{\exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_{\text{noise}})}{\exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_{\text{noise}}) + \sum_{l=1}^{N} \exp(c\boldsymbol{q}_i^\intercal \boldsymbol{k}_l)} \\ &+ \frac{\exp(m_{\text{noise}})}{\exp(m_{\text{noise}}) + \sum_{l=1}^{N} \exp(m_l)}.\end{aligned} \tag{3.18}$$

Concretely, the visually attended audio features with noise sentinel is formulated by

$$\begin{aligned}\boldsymbol{a}_i^v =& \sum_{j=1}^{N} \omega_{v\to a}(\boldsymbol{v}_i^v, \boldsymbol{a}_j^a) g_{v\to a}(\boldsymbol{a}_j^a) + \boldsymbol{a}_i^a \\ &+ \omega_{v\to a}(\boldsymbol{v}_i^v, \boldsymbol{n}_a) g_{v\to a}(\boldsymbol{n}_a).\end{aligned} \tag{3.19}$$

If the attended modality is noisy, our network can learn to ignore it by attending to the noise embedding instead. *e.g.* by setting the attention weight for the noise sentinel $\omega(\boldsymbol{v}_i^v, \boldsymbol{n}_a)$ much higher than the attention weights on the audio clips, $\omega(\boldsymbol{v}_i^v, \boldsymbol{a}_j^a)$.

3.1.4 Classifier

Let $\{\alpha_k\}_{k=1}^4$ be the set of learned weights with convex sum, $\sum_{k=1}^4 \alpha_k = 1$. The final feature representation for the i-th clip, $\boldsymbol{z}_i$, is then computed as a weighted sum of the unimodal self-attended and bimodally attended features, $\boldsymbol{z}_i = \alpha_1 \boldsymbol{v}_i^v + \alpha_2 \boldsymbol{v}_i^a + \alpha_3 \boldsymbol{a}_i^v + \alpha_4 \boldsymbol{a}_i^a$. The final bit of our framework is a two-layer network that outputs the highlight score of the i-th clip, as $s_i = \sigma\left(f_2\left(\text{ReLU}\left(f_1(\boldsymbol{z}_i)\right)\right)\right)$. Here each layer $f_i(\cdot)$, with $i \in \{1, 2\}$, contains layer-norm [53] and dropout [47] ($p = 0.5$), which is followed by a linear projection matrix W_i. The second layer $f_2(\cdot)$ specifically projects our features into a scalar, followed by the sigmoid activation $\sigma(\cdot)$ to produce the score s_i, indicating the probability of the i-th clip being a highlight.

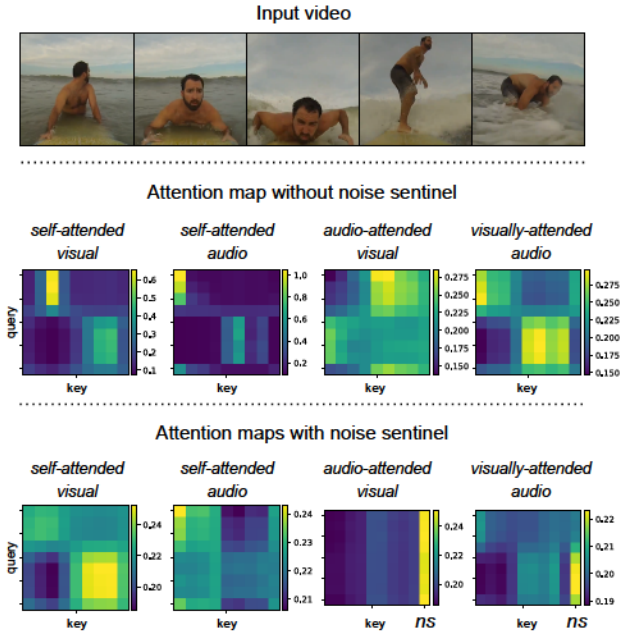

Figure 3.4: Attention maps for an example video (top row) without noise sentinel (middle row) and with noise sentinel (bottom row). *ns* denotes the attention weights placed on the noise sentinel (last column of the bimodal attention maps). This surfing video has noisy audio - the microphone constantly has water splashing against it, or is submerged in water. Consequently, our model chooses to attend largely to the noise sentinel for the audio attended visual features and visually attended audio features (bottom row) - since the audio is unreliable both as a query (there are no audio cues) and as a key (there is nothing of interest to attend to in the audio).

3.1.5 Model Learning

The highlight detection datasets are highly imbalanced, since the majority of training clips are not highlights. We address this issue by adopting a weighted binary cross entropy loss, where a positive example (highlight) carries a higher weight (w_p) than a negative example (non-highlight). Our final training loss becomes $\mathcal{L} = \sum_{i=1}^{N} -[w_p y_i \log(s_i) + (1-y_i) \log(1-s_i)]$, with $y_i \in \{0, 1\}$ the ground-truth label of the i-th clip.

3.2 Experiments

3.2.1 Datasets and Setup

We utilize three datasets, namely the YouTube Highlights dataset [1], TVSum dataset [27], as well as the Video Titles in the Wild (VTW) dataset [4]. The YouTube Highlights dataset [1] contains six different categories: dog, gymnastics, parkour, skating, skiing, and surfing. There are approximately 100 videos for each category. Following the practice of prior efforts, we train a highlight detector for each category. The TVSum dataset contains 50 videos across ten categories. We follow prior works and train a highlight detector for each category using a random 80/20 split. The Video Titles in the Wild (VTW) dataset [4] contains highlight labels, but does not have categorical information. We follow the work of [4] and adopt their split of 2,000 videos for training, 300 for validation, and 2,000 for testing.

Features: On the YouTube Highlights and TVSum datasets, we follow the work of [11] and use a 3D CNN [50] with ResNet-34 [54] backbone pretrained on the Kinetics-400 dataset [55] to obtain the visual frame-level features. Since the 3D CNN performs temporal convolution over 16 consecutive frames, we consider a 3D feature to be a part of a clip if it overlaps by at least 50% with the clip.

On the VTW dataset, we follow the original work of [4], and use a C3D network [12] pretrained on Sports-1M [56] to obtain visual features. Each video is also divided into clips of 100 frames following the prior work [4].

For all datasets, we use a PANN audio network [49] pretrained on AudioSet [57] to obtain audio features that align with the visual clips. Frame-level features are average-pooled within each clip for both audio and visual features to generate a clip-level feature.

Implementation Details: We train our model using Adam [58], with a learning rate of 5×10^{-5}. We train for 30 epochs on the YouTube and TVSum datasets, and 5 epochs on the VTW dataset. Before the attention modules, we project each modality into a vector

	RRAE (V) [3]	GIFs (V) [2]	LSVM (V) [1]	CLA (V) [8]	LM (V) [8]	MN (V) [10]	MN (AV) [10]	Trail. (V) [11]	SA (V)	Ours (AV)
dog	0.49	0.308	0.60	0.502	0.579	0.5368	0.5816	0.633	**0.649**	0.645 (↓)
gym.	0.35	0.335	0.41	0.217	0.417	0.5281	0.6165	**0.825**	0.715	0.719 (↑)
park.	0.50	0.540	0.61	0.309	0.670	0.6888	0.7020	0.623	0.766	**0.808** (↑)
ska.	0.25	0.554	0.62	0.505	0.578	0.7094	**0.7217**	0.529	0.606	0.620 (↑)
ski.	0.22	0.328	0.36	0.379	0.486	0.5834	0.5866	**0.745**	0.712	0.732 (↑)
surf.	0.49	0.541	0.61	0.584	0.651	0.6383	0.6514	**0.793**	0.782	0.783 (↑)
Avg.	0.383	0.464	0.536	0.416	0.564	0.6138	0.6436	0.691	0.705	**0.718** (↑)
Std.									±0.004	±0.006

Table 3.1: Highlight detection results (mAP) on the YouTube dataset. Our visual only model, SA (V), outperforms all prior methods, and our full audio-visual model achieves state-of-the-art performance. Best and second-best results are in bold and with underline, respectively. (↑)/(↓) indicate an improvement/decline relative to the unimodal visual baseline, SA (V). We show the modalities used for each method in brackets: (V) for visual, and (AV) for audio-visual.

of fixed-size length 512. The key, query, and value vectors all follow the same size. The constant c for all attention modules is set to 0.06. We set the positive class weight w_p to 5 for the loss term.

Evaluation Metrics: We adopt the widely-used mean Average Precision or mAP as the evaluation metric for the YouTube and VTW datasets. On the TVSum dataset, we follow prior works [8] and adopt the mAP at top-5 metric. Following prior studies [1, 2, 11], a mAP/maP at top-5 score is separately computed for every video, because a highlighted moment in one video is not necessarily more interesting than non-highlight moments in other videos; we report the average mAP over all videos. We repeat each experiment 10 times with different random seeds, and report the average performance and standard deviation over these 10 trials.

	sLSTM (V) [37]	SM (V) [35]	SG (V) [20]	LM (V) [8]	DSN (V) [21]	VESD (V) [17]	Trail. (V) [11]	MN (AV) [10]	SA (V)	Ours (AV)
VT	0.411	0.415	0.423	0.559	0.373	0.447	0.613	0.8062	0.8337	**0.8370** (↑)
VU	0.462	0.467	0.472	0.429	0.441	0.493	0.546	**0.6832**	0.6469	0.5726 (↓)
GA	0.463	0.469	0.475	0.612	0.428	0.496	0.657	0.7821	**0.8444**	0.7845 (↓)
MS	0.477	0.478	0.489	0.540	0.436	0.503	0.608	0.8183	**0.8651**	0.8605 (↓)
PK	0.448	0.445	0.456	0.604	0.411	0.478	0.591	0.7807	0.7032	**0.8009** (↑)
PR	0.461	0.458	0.473	0.475	0.417	0.485	**0.701**	0.6584	0.6749	0.6922 (↑)
FM	0.452	0.451	0.464	0.432	0.412	0.487	0.582	0.578	0.6690	**0.7003** (↑)
BK	0.406	0.407	0.417	0.663	0.368	0.441	0.647	**0.7502**	0.6808	0.7300 (↑)
BT	0.471	0.473	0.483	0.691	0.435	0.492	0.656	0.8019	0.9496	**0.9741** (↑)
DS	0.455	0.453	0.466	0.626	0.416	0.488	**0.681**	0.6551	0.6079	0.6747 (↑)
Avg.	0.451	0.461	0.462	0.563	0.424	0.481	0.628	0.7324	0.7476	**0.7627** (↑)
Std.									±0.021	±0.020

Table 3.2: Highlight detection results (top-5 mAP) on the TVSum dataset. Our visual only model, SA (V), outperforms all prior methods, and our full audio-visual model achieves state-of-the-art performance. Best and second-best results are in bold and with underline, respectively. (↑)/(↓) indicate an improvement/decline relative to the unimodal visual baseline, SA (V). We show the modalities used for each method in brackets: (V) for visual, and (AV) for audio-visual. We see an improvement in seven out of ten categories over the unimodal SA (V) baseline with our full model.

3.2.2 Baselines

We define the following baseline approaches for comparison:

- SA (A) and SA (V): In the unimodal case (audio or visual features only), we use self-attention to obtain the self-attended features, and use the same classifier architecture to classify each clip. We dub this model SA (V) for self-attended visual model, and SA (A) for self-attended audio model. The SA (V) model is directly comparable to prior works

- SA (AV)$^{\text{early}}$: We try out a self-attention based audio-visual network that concatenates the audio and visual features in an early-fusion fashion, then does self-attention and

Method	mAP
VTW (V) [4]	0.583
SA (V)	0.722 ± 0.003
Ours (AV)	$\mathbf{0.812 \pm 0.002}$

Table 3.3: Highlight detection results (mAP) on the VTW dataset. Our approach outperforms baselines, and outperforms the prior state-of-the-art method [4] by 22.9% mAP. We show the modalities used for each method in brackets: (V) for visual, and (AV) for audio-visual.

uses the same classifier to classify each clip. We call this network SA $(AV)^{early}$.

- SA $(AV)^{late}$: In addition, we try out a self-attention based audio-visual network that sums the self-attended features from each modality in a late-fusion fashion, then uses the same classifier to classify each clip. We call this network SA $(AV)^{late}$.

We also compare with state-of-the-art methods on each dataset. For each method, we use "(V)" if it only uses the visual signal, and "(AV)" if it is an audio-visual method.

3.2.3 Highlight Detection Results

YouTube Highlights: We show the main results on the YouTube dataset in Table 3.1. We achieve state-of-the-art performance on the YouTube dataset using our visual only model SA (V) model, outperforming a prior audio-visual network, MN (AV) [10]. We see a further gain of 1.3% when we use the proposed architecture. Our final architecture achieves better performance in five out of six categories, while maintaining reasonably good performance in the other category, dog.

TVSum: We show the results for the TVSum dataset in Table 3.2. Our visual-only model SA (V) outperforms the prior audio-visual state-of-the-art, and our audio-visual model improves performance by another 1.5%. We note that we see an improvement over the visual model in seven out of ten categories.

Method	TVSum	YouTube	VTW
SA (V)	$0.748^{\pm 0.02}$	$0.705^{\pm 0.004}$	$0.722^{\pm 0.003}$
SA (A)	$0.687^{\pm 0.03}$	$0.670^{\pm 0.008}$	$0.794^{\pm 0.002}$
SA $(AV)^{early}$	$0.750^{\pm 0.01}$	$0.703^{\pm 0.004}$	$0.798^{\pm 0.003}$
SA $(AV)^{late}$	$0.750^{\pm 0.02}$	$0.709^{\pm 0.004}$	$0.805^{\pm 0.003}$
Ours (AV)	$\mathbf{0.763}^{\pm 0.02}$	$\mathbf{0.718}^{\pm 0.006}$	$\mathbf{0.812}^{\pm 0.002}$

Table 3.4: Comparison to Baselines: mAP/top-5 mAP with standard deviation on TVSum, YouTube, and VTW datasets for several baselines and our full model.

VTW: We show the results for the VTW dataset in Table 3.3. Our approach outperforms the visual only baseline by a large margin. This is due in part to the importance of audio on this dataset, as shown in Table 3.4, where the audio-only baseline SA (A) outperforms its visual counterpart by 7.2% mAP. Upon further investigation, we determined that this is because the dataset contains chiefly user generated videos, where highlight-worthy moments are typically accompanied by man-made sounds such as cheering and clapping. This strengthens our belief that audio can be a useful as a stand-alone signal for highlight detection.

Comparison to Baselines: We compare our full model to the three different baselines outlined in Sec. 3.2.2. The results are shown in Table. 3.4.

Our visual-only model SA (V) outperforms its audio counterpart SA (A) on the TVSum and YouTube datasets. On the VTW dataset SA (A) outperforms SA (V) by a large margin. Qualitative analysis of the VTW dataset determined that the interesting moments in the dataset were often accompanied by sounds like clapping and cheering.

Even so, our audio-based models SA (A) often perform on-par with visual state-of-the-art methods. On the TVSum and YouTube datasets, SA (A) outperforms all but the prior state-of-the-art on the respective dataset. This lends credence to the hypothesis that audio can be a useful stand-alone signal for highlight detection.

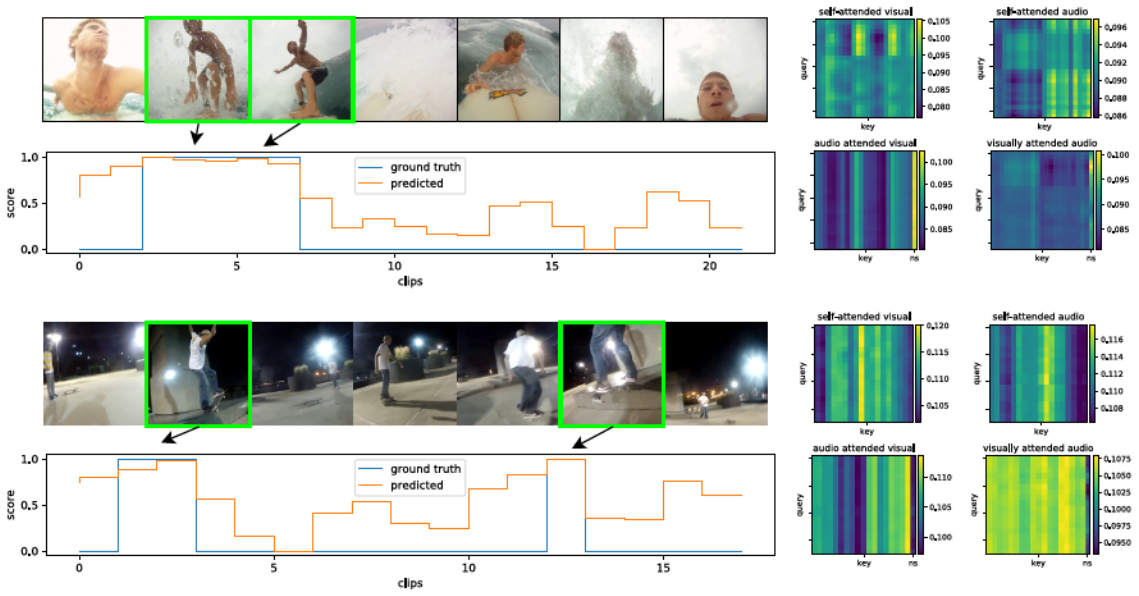

Figure 3.5: Qualitative results: Highlight moments are outlined in green, and the output of our model is plotted along with ground truth highlights below each video. The attention maps are visualized to the right of each video. The model attends to the noise sentinel *ns* for the surfing video (top), while attending to the opposite modality for the skating (bottom) video. The skating video has audio cues (cheering and skateboard noises) which our model utilizes.

Our full model outperforms the multimodal early and late fusion baselines for all three datasets. This shows the effectiveness of our bimodal attention and noise sentinel components.

Qualitative Results: We visualize the attention maps produced by our approach in Fig. 3.4. The self-attention layers produce similar attention maps. However, the attention maps of the bimodal attention layer exhibit a strong difference with and without the noise sentinel. With the noise sentinel, our model largely chooses to attend to the noise sentinel for the bimodally attended features. This particular video contains noisy audio, as the microphone is often submerged in water. Our approach learns to ignore the interaction between the modalities as the audio is not useful as a cue (query) nor as complementary information to the visual features (key, values). We present additional qualitative highlight detection results along with their attention maps in Fig. 3.5.

	Visual		Audio	
	SA (V)	Ours w/o audio	SA (A)	Ours w/o visual
YouTube	0.705	0.690 (-1.5%)	0.670	0.649 (-2.1%)
VTW	0.722	0.714 (-0.8%)	0.794	0.793 (-0.1%)

Table 3.5: Missing modality at test time: We compare our unimodal baselines to our audio-visual model with a missing modality at test time. We see small drops in performance. Our model w/o audio still outperforms the prior state-of-the-art on VTW by 13.1% [4], and performs almost as well as the prior state-of-the-art on YouTube [11] (69.0% vs. 69.1%).

3.2.4 Ablation Studies

Missing Modality during Testing: During training, our approach has full access to both visual and audio modalities. Meanwhile at testing, we may sometimes encounter cases where only one modality is available. This motivates us to investigate the situation of missing a modality at test time. Here, the missing modality is simulated by feeding in an input sequence of all-zeros in place of the missing modality. Table 3.5 shows the results.

On the YouTube dataset, a performance drop is observed relative to the unimodal SA (V) and SA (A) models of 1.5% and and 2.1%, respectively, which is to be expected. Even with this performance dip, our approach performs on par with the previous state-of-the-art on with a visual-only input: we achieve a mAP of 69.0% while the previous state of the art is 69.1%. We see much smaller drops in performance on the VTW dataset. Using only visual features at test-time, our performance is 0.8% worse than the SA (V) visual baseline. When we use audio features only, we note a performance drop of only 0.1%. For the VTW dataset, we outperform the prior state-of-the-art by 13.1% when we are missing audio, and by 21% when we are missing visual features. Overall, our audio-visual approach is

visual feat.	YouTube → VTW	VTW → YouTube
C3D	0.634	0.616
3D ResNet	0.623	0.631

Table 3.6: Cross-dataset testing results: We train on one dataset and test on the other, and outperform the majority of prior works.

robust to missing modalities. As most videos come with both audio and visual features, our method is expected to outperform visual-only and audio-only models, even when a modality is missing in some videos.

Cross-Dataset Testing: In this section, we investigate the transfer performance of our model across datasets. Since we use different visual features to train our main models for each dataset (3D ResNet-34 for YouTube, and C3D for VTW), we show two possibilities for each experiment in Table 3.6. For example, the first row of YouTube → VTW indicates that we use C3D visual features to train a model on YouTube, and test on the VTW dataset.

When training on VTW and testing on YouTube, our method outperforms all prior visual-only methods except for Trailers [11] that are trained and tested on the YouTube dataset. Our cross-dataset performance from YouTube to VTW shows similar results, where it outperforms the results reported in [4] on the VTW dataset by 5.1% mAP, using the same visual features.

Impact of Disentangled Attention: In this experiment, we analyze the impact of disentangled attention compared with classic self-attention, where the attention score is formulated as $\omega_{v\to v}(\boldsymbol{v}_i, \boldsymbol{v}_j) = \text{softmax}(c\hat{\boldsymbol{q}}_{v\to v}(\boldsymbol{v}_i)^\intercal \hat{\boldsymbol{k}}_{v\to v}(\boldsymbol{v}_j))$ for self-attention and similarly for our bimodal attention layers. We show the result in Table 3.7. We see that disentangled attention improves our results for both datasets.

Impact of Noise Sentinel: In this experiment, we study the impact of the noise sentinel on the overall performance. We show the result in Table 3.8. The result demonstrates that using the noise sentinel improves the performance on both datasets.

Attention mechanisms	VTW	YouTube
Classic self-attention	0.808	0.705
Disentangled attention (ours)	**0.812**	**0.718**

Table 3.7: Ablation study on disentangled attention: We achieve the best result with the disentangled attention.

Noise sentinel	VTW	YouTube
Without noise sentinel	0.804	0.716
With noise sentinel (ours)	**0.812**	**0.718**

Table 3.8: Ablation study on noise sentinel: We achieve superior performance with the noise sentinel.

3.3 Summary

We proposed an audio-visual framework for video highlight detection. Our architecture models the relationship between audio and visual information through a bimodal attention layer to produce fused representations. We also introduced an adaptive noise sentinel mechanism that "looks away" from a noisy audio or visual modality. We empirically showed that our framework achieves superior performance on well-known benchmark datasets. While we focus on highlight detection, the proposed techniques could benefit other audio-visual tasks such as audio-visual speech recognition, action localization and video summarization.

Chapter 4

Contrastive Learning for Unsupervised Video Highlight Detection

In this chapter we consider the problem of unsupervised video highlight detection. Existing efforts in highlight detection are mostly fully-supervised, requiring humans to manually label the interesting moments in videos. This is a very expensive and time intensive process. Recent weakly supervised methods [8, 10] alleviate the heavy demand for manual annotations, but nonetheless collect large-scale external to compensate for the lack of ground-truth annotations. For example, the work of [8] employs 10 million Instagram videos to train their model.

On the other hand, recent advances in contrastive learning have helped close the gap between unsupervised models and their supervised counterparts [48, 59]. The goal of contrastive learning is to learn an unsupervised representation of some data that is useful for downstream tasks. For images, the task is usually classification. In the image domain, a model is trained to map two randomly transformed versions of the same image (*e.g.* random cropping, color distortion), closer together in an embedding space relative to other images that are also randomly transformed [59]. Thus, we can view contrastive learning as a clustering task in which the aim is to form a cluster for each image where the samples within each cluster are randomly transformed versions of the original image.

While visual transformations are straightforward, it is unclear what kinds of transformations to apply in discrete domains such as language. Textual transformations could include

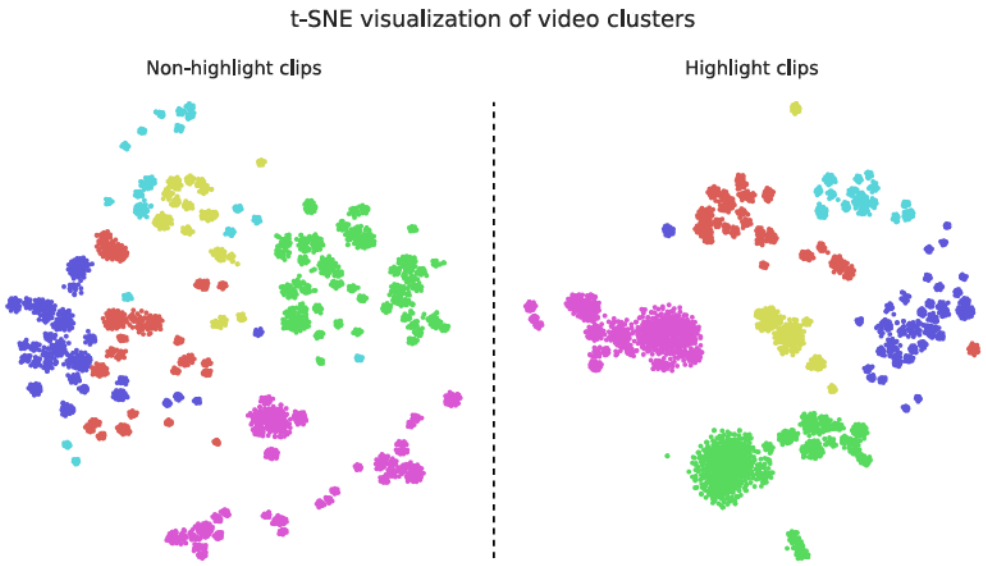

Figure 4.1: Highlight clips form better clusters for contrastive learning: we use t-SNE [60] to visualize a video and its nearest neighbors using non-highlight clips and highlight clips. Each point represents a clip, and each color represents a video. We see that poor clusters in the non-highlight clip embedding space are clearer in the highlight clip embedding space. Thus, our model learns to pick highlight clips to do well on the contrastive learning task.

word deletion, replacement with a synonym, or even applying back-translation to obtain a different phrase with the same meaning. Recent work [48] has found that dropout [47] can be used as a random transformation to learn superior unsupervised sentence embeddings through contrastive learning.

Inspired by this idea, we propose our own attention-based network that directly picks clips in order to produce a video embedding. We apply dropout within the network as our transformation, and learn to map two embeddings of the same video with a different series of dropouts closer than to embeddings of other videos.

While the end-goal in the aforementioned works in contrastive learning is to produce a representation of the entire image or a sentence, we are much more interested in which parts of the video our model picks to produce this representation. Each video contains highlight and non-highlight clips. We posit that highlight clips contain more information about each video and thus our model learns to pick highlight moments to perform well on

the contrastive learning task.

Let us motivate this claim through the example in Fig. 4.1. As previously stated, we are essentially learning to cluster a video close to itself under random dropout perturbations. We claim that highlight clips contain more information about the video itself, and thus we form better clusters if our model picks highlight clips to produce a video embedding. In this figure, we show the same set of videos represented by non-highlight clips (left) and highlight clips (right) using t-SNE [60]. Each video is shown using a different color, and each point represents a clip. The non-highlight clips do not form good clusters: the clusters often overlap, and each cluster is also not compact. On the other hand, we can clearly differentiate between the different videos when using highlight clips. This means that in order to do well on the contrastive learning task, our model will learn to pick highlight clips. We also empirically investigate the quality of the clusters in Section 4.2.3.

We empirically evaluate our model on three widely-used highlight detection benchmarks, and demonstrate the superior performance of our approach. We significantly outperform the state-of-the-art methods.

4.1 Approach

As illustrated in Fig. 4.2, our approach is based on contrastive learning. In the contrastive learning task, we learn to map a transformed version of a sample close to itself relative to other samples in an embedding space. In order to do this, we use different dropout masks within our network to produce a transformed version of each video. Our method then generates an attention score for each clip to compute a weighted sum of the clips. This weighted sum is used to represent the entire video for contrastive learning. Our key hypothesis is that our network will learn to pick highlight clips in order to solve the task. In Fig. 4.3, we show our model inference at test time for highlight detection. We use the attention scores from the clip selector directly as our highlight scores.

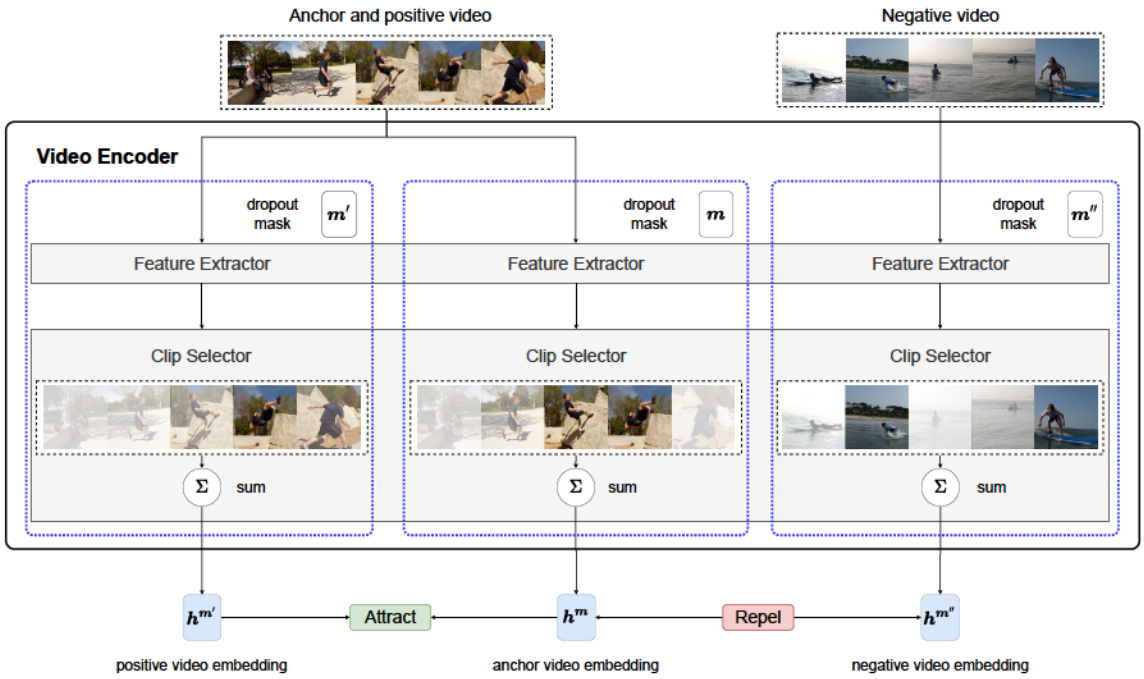

Figure 4.2: Contrastive Learning Framework for training: we use a video encoder model to embed an input video. We obtain two different embeddings $\boldsymbol{h}^{m}$ and $\boldsymbol{h}^{m'}$ of a video by using different dropout masks $\boldsymbol{m}$ and $\boldsymbol{m}'$, respectively. These two embeddings will serve as the positive video embedding, and an anchor video embedding. We also obtain a negative video embedding $\boldsymbol{h}^{m''}$ by using a different video as a negative video. Our objective is to increase the similarity between the anchor embedding $\boldsymbol{h}^{m}$ and positive embedding $\boldsymbol{h}^{m'}$ while decreasing the similarity between the anchor embedding $\boldsymbol{h}^{m}$ and the negative embedding $\boldsymbol{h}^{m''}$. In practice, we do this using the contrastive loss defined in Eq. 4.14, which uses multiple negative video embeddings in a batch. In this framework, we can see that the clip selector learns to pick the most informative clips in the video.

4.1.1 Generic Attention Layer

We define a generic attention layer to simplify notation. The generic attention layer is a modified version of the scaled dot product attention of [51] as illustrated in Fig. 4.4. Consider two sequences of d-dimensional vectors $\boldsymbol{X} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_K\}$ and $\boldsymbol{Y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_L\}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ $(i = 1, 2, ..., K)$ and $\boldsymbol{y}_j \in \mathbb{R}^d$ $(j = 1, 2, ..., L)$. We define an attention layer that queries $\boldsymbol{Y}$ using $\boldsymbol{X}$. The attention layer returns an intermediate attention map $\alpha \in \mathbb{R}^{K \times L}$ that captures the degree of dependence between $\boldsymbol{x}_i$ and $\boldsymbol{y}_j$, and computes the attended features $\boldsymbol{Z} = \{\boldsymbol{z}_1, \ldots \boldsymbol{z}_K\}$. We define the following general form for notational

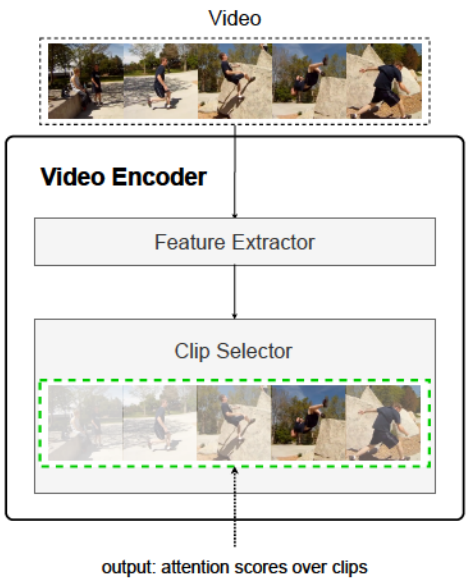

Figure 4.3: Model inference at test time: we don't make use of the embeddings and simply use the attention scores over the clips produced by our clip selector as our output highlight scores.

convenience:

$$\alpha, \boldsymbol{Z} = \text{Attn}(\boldsymbol{X}, \boldsymbol{Y}). \tag{4.1}$$

The details of Eq. (4.1) are as follows. First, we define the query, key, and value linear transformations:

$$\boldsymbol{q}_i = \boldsymbol{W}_q \boldsymbol{x}_i, \ \boldsymbol{k}_j = \boldsymbol{W}_k \boldsymbol{y}_j, \ \boldsymbol{v}_j = \boldsymbol{W}_v \boldsymbol{y}_j, \tag{4.2}$$

where $\boldsymbol{W}_q \in \mathbb{R}^{d \times d}, \boldsymbol{W}_k \in \mathbb{R}^{d \times d}, \boldsymbol{W}_v \in \mathbb{R}^{d \times d}$ are model parameters of the linear transformations. We formulate the attention map α using the dot product between the query and key, and normalize over the keys (j) using the softmax function.

$$\alpha_{ij} = \text{softmax}(\boldsymbol{q}_i \cdot \boldsymbol{k}_j). \tag{4.3}$$

We define $\boldsymbol{z}_i$ as a weighted sum of the values $\boldsymbol{v}_j$:

$$\boldsymbol{z}_i = \sum_{j=1}^{N} \alpha_{ij} \boldsymbol{v}_j. \tag{4.4}$$

We also include a residual connection, followed by layer normalization [53], that is:

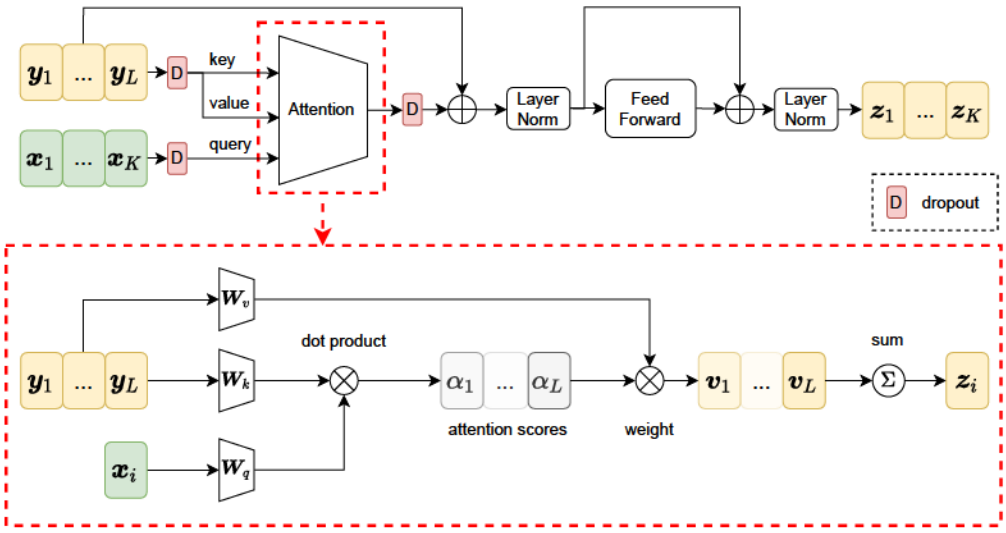

Figure 4.4: Generic Attention Layer: our generic attention layer takes in a query sequence $\boldsymbol{x}_i,\ i = 1, ..., K$, and a key/value sequence $\boldsymbol{y}_j,\ j = 1, ..., L$. It then produces an attended feature $\boldsymbol{z}_i$ for each query $\boldsymbol{x}_i$. We apply dropout to both input sequences, and apply dropout after computing the attention between the two sequences. This is then followed by a residual connection and a layer normalization layer. Finally, we have a feed-forward layer with a residual connection, before applying another layer normalization layer at the end. Residual connections are used only when $L = K$.

$$\boldsymbol{z}_i \leftarrow \text{LN}(\boldsymbol{z}_i + \boldsymbol{y}_i), \tag{4.5}$$

where LN() indicates layer normalization.

After this, we feed the output to a feed-forward network that also includes a residual connection and layer normalization. The feed-forward network is simply two linear layers with a ReLU non-linearity in between. Thus our final attended features are:

$$\boldsymbol{z}_i \leftarrow \text{LN}(\boldsymbol{z}_i + \text{FF}(\boldsymbol{z}_i)), \tag{4.6}$$

where $\text{FF}(\boldsymbol{z}_i) = \max(0, \boldsymbol{z}_i\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$ and $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d\times d}$, $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^d$ are the weights and biases, respectively, of the feed-forward network.

The attention layer allows us to capture the interaction between the two sequences. If our query and key sequences have differing lengths (i.e. $K \neq L$) we are unable to use Eq. 4.5, and simply remove the residual connection and perform layer normalization directly on $\boldsymbol{z}_i$.

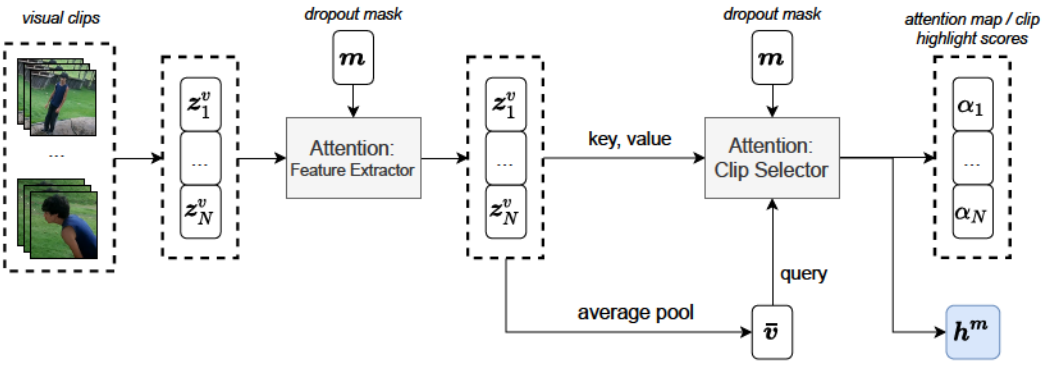

Figure 4.5: Video Encoder Architecture: We break a video into clips of some fixed number of frames (*e.g.*100 frames) and obtain N clips. For each clip j we obtain features from a pre-trained feature extractor to get $\boldsymbol{v}_j$. Then we use our feature extractor attention layer to obtain higher level self-attended features. We then average the self-attended features to obtain one query $\bar{\boldsymbol{v}}$, and produce a weighted sum of the self-attended features using our clip selector attention layer. In each attention layer, we apply a dropout mask $\boldsymbol{m}$. While this dropout mask is not the same for each attention layer, we simply use $\boldsymbol{m}$ to denote the entire sequence of dropouts applied in a forward pass for one video. The final output is the set of attention scores α_j which serve as our highlight scores s_j for each clip, and a video embedding $\boldsymbol{h}^m$ which we use for the contrastive learning task.

4.1.2 Video Encoder

We split a video into N fixed-length clips. The content of the clips is then represented using N vectors of dimension d_v, $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$. Here $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ and d_v is the dimension of the clip feature. These features are extracted using pre-trained visual feature extractors. Our goal is then to produce highlight scores $S = \{s_1, \ldots, s_N\}$ for each clip in the video.

Feature Extractor: In video highlight detection, it is important to capture the relationship between different clips in a video, as it is difficult to reason whether a clip is a highlight without any context. In order to capture this relationship, we use an attention layer. First, the video features $\boldsymbol{v}_i \in \mathbb{R}^{d_v}$ are projected to a fixed embedding of size d,

$$\boldsymbol{z}_i^v = \boldsymbol{W}_{\mathcal{V}} \boldsymbol{v}_i, \tag{4.7}$$

where $\boldsymbol{W}_{\mathcal{V}} \in \mathbb{R}^{d \times d_v}$. We use $\boldsymbol{Z}^v$ to denote the set of features of a video, i.e.

$$\boldsymbol{Z}^v = \{\boldsymbol{z}_1^v, \boldsymbol{z}_2^v, ..., \boldsymbol{z}_N^v\}. \tag{4.8}$$

Then we perform self-attention to update the features:

$$\boldsymbol{Z}^v \leftarrow \text{Attn}_f(\boldsymbol{Z}^v, \boldsymbol{Z}^v). \tag{4.9}$$

Note that the self-attention is a specific case of our generic attention layer from Sec. 4.1.1, where the sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ are identical.

Clip Selector: The clip selector picks important clips within the video using a soft-mechanism. While there are many choices for how to define this, we simply use another attention layer. Simply put, we define the query as the average of the features obtained from the feature extractor, and attend to the features $\boldsymbol{Z}^v$. This will return a single embedding at the output of our attention layer since the number of queries K is 1:

$$\bar{\boldsymbol{v}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{z}_i^v, \tag{4.10}$$

$$\alpha, \{\boldsymbol{h}\} = \text{Attn}_{cs}(\{\bar{\boldsymbol{v}}\}, \boldsymbol{Z}^v). \tag{4.11}$$

This gives us our final video embedding $\boldsymbol{h} \in \mathbb{R}^d$, and the attention scores α that denote the contribution of each clip to the video embedding. Through this mechanism, our model learns to pick highlight clips in order to distinguish the video from other videos within the dataset. We denote our video encoder as $\boldsymbol{h} = F(\boldsymbol{V})$, and use the attention score α_j directly as our highlight score s_j for clip j.

4.1.3 Contrastive Learning

We illustrate the contrastive learning framework in Fig. 4.2. In the contrastive learning task, we learn to map a transformed version of a sample close to itself relative to other samples in an embedding space. In the case of images, it is easy to think of different transformations. Common transformations include random cropping, random flips, and color distortions. However, such data transformations are difficult to apply to discrete modalities such as language [48], and in our case, vector representations of video clips obtained from a pre-

trained feature extractor. To combat this issue, we adopt the SimCSE framework for our task and use dropout as our transformation [48].

Dropout: Dropout was first introduced in the work of [47] to prevent neural networks from overfitting. In dropout, an activation inside the neural network is randomly set to zero with some probability p. Following SimCSE [48], we opt instead to use dropout as a transformation: given two different dropout masks within our network, we will obtain two different video embeddings. This is the analog to images, where we obtain two different embeddings for an image if apply different transformations (e.g. random crop, random flip) to the image before we feed it into the network.

In our model, we apply dropout in two places for each attention layer: first, we apply dropout to the features before we have computed the dot product attention. That is,

$$\boldsymbol{x}_i \leftarrow \text{Dropout}(\boldsymbol{x}_i),\ \boldsymbol{y}_i \leftarrow \text{Dropout}(\boldsymbol{y}_i). \tag{4.12}$$

In addition, we apply dropout to the output of the dot-product attention before layer normalization and the residual connection. Thus Eq. 4.5 becomes:

$$\boldsymbol{z}_i \leftarrow \text{LN}(\text{Dropout}(\boldsymbol{z}_i) + \boldsymbol{y}_i). \tag{4.13}$$

We use $\boldsymbol{m}$ to denote the entire sequence of dropout masks applied within the network for an input video. Given this mask, we denote the embedding produced by our video encoder as $\boldsymbol{h}^{\boldsymbol{m}} = F(\boldsymbol{V}, \boldsymbol{m})$.

Contrastive Loss: We adopt the common softmax formulation of the contrastive loss [48, 59]. For a video l, we apply two different dropout masks $\boldsymbol{m}$ and $\boldsymbol{m'}$ to produce the anchor video embedding $\boldsymbol{h}_l^{\boldsymbol{m}} = F(\boldsymbol{V}_l, \boldsymbol{m})$ and positive video embedding $\boldsymbol{h}_l^{\boldsymbol{m'}} = F(\boldsymbol{V}_l, \boldsymbol{m'})$, respectively. We randomly sample other videos in the dataset k to form the negative video embeddings $\boldsymbol{h}_k^{\boldsymbol{m}'_k}$. Our task is then to increase the similarity between the anchor and positive embeddings and decrease the similarity between the anchor and negative embeddings. Concretely, we formulate the loss for video l as follows:

$$\mathcal{L}_l = -\log \frac{\exp(\text{sim}(\boldsymbol{h}_l^{m_l}, \boldsymbol{h}_l^{m'_l})/\tau)}{\sum_{k=1}^{B} \exp(\text{sim}(\boldsymbol{h}_l^{m_l}, \boldsymbol{h}_k^{m'_k})/\tau)}. \tag{4.14}$$

Here, B denotes the batch size, τ is a temperature parameter, and $\text{sim}(\boldsymbol{h}_1, \boldsymbol{h}_2)$ denotes the cosine similarity $\frac{\boldsymbol{h}_1^\intercal \boldsymbol{h}_2}{||\boldsymbol{h}_1|| \cdot ||\boldsymbol{h}_2||}$. This loss is averaged over an entire batch.

4.2 Experiments

4.2.1 Datasets and Setup

We empirically evaluate our model on three benchmark datasets, namely the YouTube Highlights dataset [1], TVSum dataset [27], as well as the Video Titles in the Wild (VTW) dataset [4]. The YouTube Highlights dataset [1] contains videos of six different categories: dog, gymnastics, parkour, skating, skiing, and surfing. For each category, there are approximately 100 videos. We pool all the videos together, and train a general model for all categories. The TVSum dataset contains 50 videos across ten categories. Just like the YouTube dataset, we pool all the videos together and train a general model for all categories. Following prior works, we use 80% of the dataset for training, and 20% for testing [19, 21]. The Video Titles in the Wild (VTW) dataset [4] is a general dataset containing unedited videos captured using mobile devices. The videos are not constrained to specific categories, so that this dataset can be considered "in-the-wild". We follow the work of [4] and adopt their split of 2,000 videos for training, and 2,000 for testing.

Features: On the YouTube Highlights and TVSum dataset, we follow prior works [8, 11] and use a 3D CNN [50] with ResNet-34 [54] backbone pretrained on the Kinetics-400 dataset [55] to obtain the visual frame-level features. Since the 3D CNN performs temporal convolutions over 16 consecutive frames, we consider a 3D feature to be a part of a clip if it overlaps by at least 50% with the clip.

On the VTW dataset, we follow the original work of [4] and use a C3D network [12] pretrained on the Sports-1M [56] dataset to obtain visual features. Each video is also divided into clips of 100 frames following the original work [4].

Method	RRAE [3]	LM-A [8]	LM-S [8]	MN [10]	Ours
External data	✓	✓	✓	✓	✗
dog	0.49	0.519	0.579	0.5368	**0.6057**
gym.	0.35	0.435	0.417	0.5281	**0.7109**
park.	0.50	0.650	0.670	0.6888	**0.7422**
ska.	0.25	0.484	0.578	**0.7094**	0.4976
ski.	0.22	0.410	0.486	0.5834	**0.6820**
surf.	0.49	0.531	0.651	0.6383	**0.6852**
Avg.	0.383	0.505	0.564	0.6138	**0.6539**

Table 4.1: Comparison to unsupervised and weakly supervised methods on the YouTube dataset (mAP): our model outperforms all prior methods by a large margin. To our knowledge, no prior unsupervised or weakly supervised work exists that dose not make use of external data on the YouTube dataset.

Implementation Details: We train our model using the Adam optimizer [58] with a learning rate of 1e-5 for 10 epochs on each dataset. We simply evaluate the model at the last epoch for the given dataset. We set our contrastive batch size to 32 videos. Before the attention modules, we set the embedding size d to 512. The key, query, and value vectors all follow the same size. We set the temperature τ to 0.1, and our dropout rate p to 0.1 for all datasets.

Evaluation Metrics: We use mean Average Precision (mAP) to evaluate our model. Following prior studies [1, 2, 11], we compute the mean Average Precision separately for every video, because a highlight in one video is not necessarily more interesting than non-highlights in other videos; we report the average mAP over all videos.

Comparison Methods: We compare our method to several state-of-the art methods on the YouTube dataset: RRAE [3], GIFs [2], LSVM [1], LM-A, LM-S [8], MN [10], Trailers [11], and Attn [61]. Prior weakly supervised methods, RRAE, LM-A, LM-S, and MN rely

Method	MBF [18]	CVS [19]	SG [20]	DSN [21]	Ours
External data	✗	✗	✗	✗	✗
mAP	0.345	0.372	0.462	0.424	**0.5276**

Table 4.2: Comparison to unsupervised and weakly supervised methods on TVSum dataset (mAP at top-5) that don't use external data. Our model achieves the best performance by a large margin.

Method	VTW [4]	Attn. [61]	Ours
Supervised	✓	✓	✗
mAP	0.583	0.722	0.6090

Table 4.3: Highlight detection results (mAP) on the VTW dataset: We achieve reasonable performance compared to a fully supervised state-of-the-art network, and outperform one prior supervised work.

on the collection of large external datasets, while our model is trained on the YouTube dataset itself.

On the TVSum dataset, we compare our method to prior unsupervised and weakly supervised methods: MBF [18], CVS [19], SG [20], VESD [17], LM-S [8], MN [10], LM-A [8], and DSN [21]. Of these works VESD, LM-A, LM-S, and MN make use of external data.

While we have fewer baselines on the VTW dataset, we compare our unsupervised model to prior supervised works [4, 61], and show competitive performance.

4.2.2 Unsupervised Highlight Detection

YouTube: We compare our model to other unsupervised and weakly supervised methods in Table 4.1. We outperform the prior state-of-the-art in weakly supervised highlight detection by 4%. Our performance is especially notable since we do not use any external data, unlike the compared works which all collect large amounts of external data to train their

Method	VESD [17]	LM-A [8]	LM-S [8]	MN [10]	Ours
External data	✓	✓	✓	✓	✗
mAP	0.423	0.524	0.563	0.6979	0.5276

Table 4.4: Comparison to unsupervised and weakly supervised methods on TVSum dataset (mAP at top-5) that use external data. Even without external data our model outperforms two of methods.

algorithms. To our knowledge no prior weakly or unsupervised work exists that does not collect external data to train their model on the YouTube dataset.

We compare our model to prior supervised methods in Table 4.5. We outperform two prior supervised works, and achieve a performance only 5.1% below the supervised state-of-the-art, Attn [61]. This proves the efficacy of our model.

TVSum: We compare our model to other unsupervised and weakly supervised models on the TVSum dataset that don't use external data in Table 4.2. Our model outperforms the state-of-the-art significantly by 6.5%.

We also compare our method to models that make use of external data in Table 4.4. Our model achieves competitive performance. In particular, we outperform VESD and LM-A, and achieve a surprisingly competitive performance to LM-S. LM-A and LM-S are trained on 10 million Instagram videos, and LM-S is the category-specific version of LM-A. Our model also achieves competitive performance to LM-A and LM-S despite not using external data. The TVSum dataset itself is quite small, with only fifty videos in total for training and testing, so the gap in performance is to be expected.

VTW: We present our unsupervised learning results for VTW in Table 4.3. Note that even though our model is unsupervised and does not require ground-truth highlights on training videos, it performs better than a prior *supervised* work VTW [37], while maintaining reasonable performance relative to the supervised state-of-the-art, Attn [61]. We outperform the prior supervised work by 2.6%, and underperform relative to the super-

Method	GIFs [2]	LSVM [1]	Trail. [11]	Attn. [61]	Ours
Supervised	✓	✓	✓	✓	✗
dog	0.308	0.60	0.633	0.649	0.6057
gym.	0.335	0.41	0.825	0.715	0.7109
park.	0.540	0.61	0.623	0.766	0.7422
ska.	0.554	0.62	0.529	0.606	0.4976
ski.	0.328	0.36	0.745	0.712	0.6820
surf.	0.541	0.61	0.793	0.782	0.6852
Avg.	0.464	0.536	0.691	0.705	0.6539

Table 4.5: Comparison to supervised methods on the YouTube dataset (mAP): Our model underperforms the supervised state-of-the-art, Attn [61], by only 5% mAP. Their work uses full supervision and they also train a separate model for each category.

vised state-of-the-art by 11.3%. This is quite a strong performance from a model trained only to distinguish between videos by directly picking clips.

VTW is an interesting dataset since it contains unedited "in-the-wild" videos that are captured through personal recording devices such as camcorders and smartphones. Our model works well on this highly unconstrained dataset, demonstrating the generalization ability of our model to "in-the-wild" videos.

4.2.3 Why Does It Work?

In essence, we learn to cluster a video embedding close to itself under small dropout perturbations. If our model learns to pick non-highlight moments, we posit that the quality of the clusters will be worse, and thus our performance on the contrastive learning task is also worse.

To see why this is the case, let us assume that we have a hypothetical model that always picks non-highlight clips, and another hypothetical model that always picks highlight clips.

	YouTube		TVSum		VTW	
Highlight clips	✗	✓	✗	✓	✗	✓
Intra-cluster distance (↓)	0.0954	**0.0888**	0.1297	**0.1225**	0.1010	**0.0901**
Inter-cluster distance (↑)	0.2325	**0.2597**	0.0905	**0.0981**	0.8241	**0.8284**

Table 4.6: Non-highlight clips form poorer clusters: a model that learns to pick highlight clips achieves better clustering than a model that learns to pick non-highlight clips. Highlight clips form compact clusters (small intra-cluster distance) that are farther away from other clusters (large inter-cluster distance). (↓) indicates smaller is better, while (↑) indicates bigger is better.

Let us assume that each model picks a clip from the output of our feature extractor exactly (the attention map α of the clip selector is 1 at some index i and 0 everywhere else). For this part, we operate directly on the clip representations at the output of the feature extractor and do not apply any other transformations.

We sample the same number of non-highlight and highlight clips from each video, and run each model twenty times with different dropout masks for the chosen non-highlight/highlight clips. In order to perform well on the contrastive learning task, the embeddings output by each model should form a cluster for each video that has a small intra-cluster distance (embeddings belonging to the same video should be close together), and a large inter-cluster distance (embeddings belonging to different videos should be far apart).

We use the cosine distance $\text{dist}(\boldsymbol{h}_1, \boldsymbol{h}_2) = 1 - \frac{\boldsymbol{h}_1 \cdot \boldsymbol{h}_2}{||\boldsymbol{h_1}|| \cdot ||\boldsymbol{h_2}||}$ as our distance measure. For a video k we sample N highlight or non-highlight clips to form a video cluster. We define the centroid as simply the average of all the sampled clip embeddings $\bar{\boldsymbol{h}}_k = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{h}_{ik}$. The intra-cluster distance is then defined as the mean of the cosine distance between the centroid and the clip embeddings i.e. $\frac{1}{N} \sum_{i=1}^{N} \text{dist}(\boldsymbol{h}_{ik}, \bar{\boldsymbol{h}}_k)$. The intra-cluster distance is a measure of how compact the cluster is. Thus, a smaller number is better.

Now let us define the inter-cluster distance. The inter-cluster distance is a measure of

Dropout	YouTube	TVSum	VTW
✗	0.6287	0.4887	0.5606
✓	**0.6539**	**0.5276**	**0.6090**

Table 4.7: Effect of dropout on performance. We use a dropout rate p of 0.1 for all three datasets.

how well video clusters are separated from each other, thus a larger number is better. We define this as the mean distance between the centroid of a video cluster k to the centroid of other clusters j. Concretely, we can write this out as $\frac{1}{M}\sum_{j=1,j\neq k}^{M} \text{dist}(\bar{\boldsymbol{h}}_k, \bar{\boldsymbol{h}}_j)$. M is the total number of videos. We compute the intra-cluster and inter-cluster statistics for each video, and report the average of each metric across all videos.

We show our results in Table 4.6. We can see that choosing non-highlight clips leads to worse clustering in the embedding space. The intra-cluster distance (distance between embeddings for the same video) is higher for non-highlight clips, and lower for highlight clips. This means that highlights form a tighter cluster in the embedding space. On the other hand, the inter-cluster distance is higher when using highlight clips compared when we use non-highlight clips. Highlights form clusters that are well separated. This is a desirable property for contrastive learning.

This shows that highlight clips are more informative about the video itself. Thus our model learns to pick highlight clips in order to perform well on the contrastive learning task.

4.2.4 Effect of dropout on performance

We investigate the effect of dropout in this section, and show the results in Table. 4.7. We note that dropout improves our performance by noticeable margins (2-4% mAP). Without dropout, the contrastive learning task is simpler: the cosine similarity between the anchor video embedding $\boldsymbol{h}_l^{\boldsymbol{m}}$ and the positive video embedding $\boldsymbol{h}_l^{\boldsymbol{m}'}$ for the dropout masks $\boldsymbol{m}$ and $\boldsymbol{m}'$ would simply be 1 because $\boldsymbol{m} = \boldsymbol{m}'$. However, our model would still have to learn to

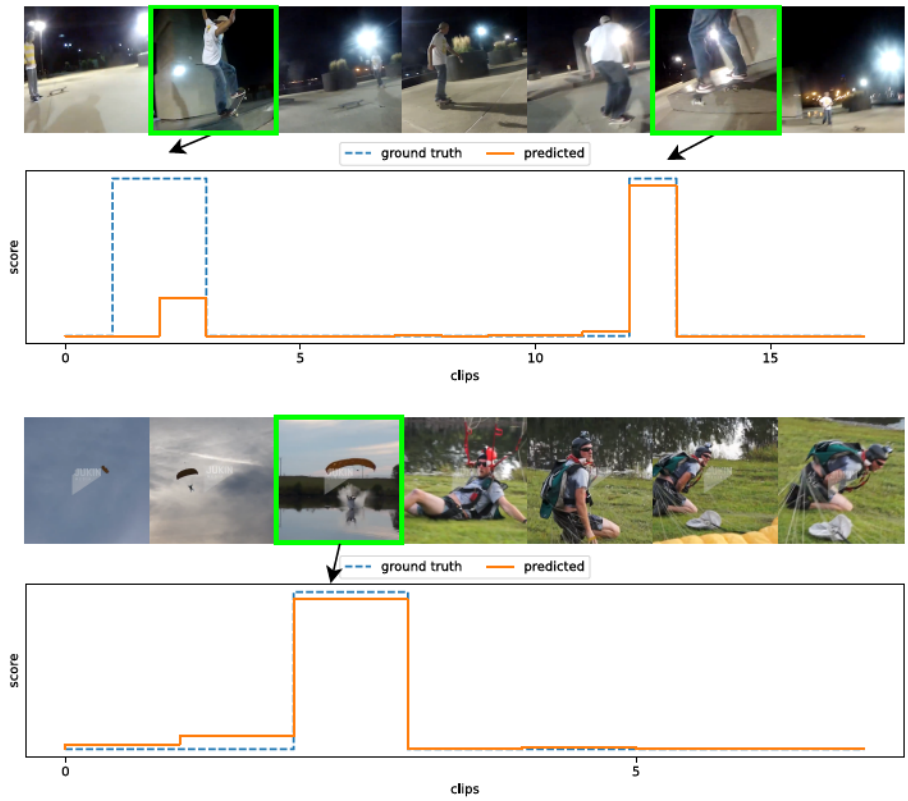

Figure 4.6: Qualitative results: we show some qualitative examples. Ground truth highlight moments are outlined in green. In the skateboarding video (top), we see two highlights. While our model chooses to focus mostly on the second highlight, we see that it detects the first as well. In the second video of somebody crashing while paragliding, our model correctly identifies the highlight moment.

decrease the similarity between the anchor embedding $\boldsymbol{h}_l^m$ and the negative embeddings in the batch $\boldsymbol{h}_k^{m'_k}$. As such, our results are still competitive to prior works for all three datasets.

4.2.5 Qualitative Results

We show qualitative results in Fig. 4.6 and see that our model accurately identifies the highlight moments. In the first video, a skateboarder does two tricks which are outlined in green. In the second video, a man who is paragliding skims the water and nearly crashes. The moment where he skims the water is the highlight moment.

4.3 Summary

In this chapter, we presented a simple unsupervised method for highlight detection. Our model works simply by learning to distinguish between a dropout transformed version of a video from other videos by picking highlight clips within the video. We do not collect large external data, unlike many prior works in unsupervised and weakly supervised highlight detection. Empirically, we showed state-of-the-art performance on three benchmark highlight detection datasets.

Chapter 5

Conclusion

Video highlight detection aims to find interesting moments in videos. It is an important task due to the sheer amount of video content being generated online. It has a broad range of downstream applications including video summarization, recommendation, editing, and browsing. In this thesis, we have developed two novel methods for video highlight detection that improve upon the state-of-the-art.

In Chapter 3, we presented a supervised model that makes use of audio for video highlight detection. Our model captures the interaction between video and audio, and also has an automatic mechanism for ignoring either modality when it is noisy. Our model achieved superior results, leading us to conclude that highlight detection benefits from the use of audio.

In Chapter 4, we proposed an unsupervised method for video highlight detection. Unlike most existing approaches which depend on manually annotated training data, our method does not make use of any labels. We also don't collect external video data for training our model, unlike many existing weakly supervised and unsupervised methods. We achieved superior results showing the effectiveness of our unsupervised highlight detection framework.

There are several different avenues of exploration in highlight detection. For example, since we have shown that audio can be useful for video highlight detection, methods that perform test-time model adaptation are of interest [62, 63]. The core idea is to adapt the

weights of the model at test-time to a specific instance to improve the performance on that instance for our main task. This is often done by updating the weights of the model through an auxiliary task. In our case, the auxiliary task could be tasks such as audio prediction from video, or audio-visual alignment. Another interesting direction is to incorporate text as an input in addition to vision and audio for highlight detection since most videos online come with titles or a short description of what is happening in the video.

In terms of unsupervised video highlight detection, we can consider methods that bridge the gap between unsupervised and supervised methods. The goal would be to train on a large number of videos in an unsupervised manner, then fine-tune on a few labeled samples to perform on par with supervised networks in video highlight detection. This approach has attracted a lot of attention in recent years in the image classification literature [46, 59], so it could be of great interest to apply a similar framework to video understanding tasks.

In terms of the highlight detection task itself, we can consider several different variants: we could apply highlight detection to 360 degree videos. We can also try applying video highlight detection directly to compressed videos. While existing works all operate on RGB frames, most visual features are redundant since there is very little change from one frame to the next. Existing compression formats such as MPEG-4 rely on this redundancy and only embed the difference between consecutive frames. Existing work in action recognition [64] suggests that we can achieve superior performance on video based tasks while also saving computational time by directly operating on this compressed format.

We hope that the methods presented in our work can inspire future methods in video based tasks such as video action recognition, action localization, and video summarization. In this thesis, we have taken another step towards understanding videos the same way we do as humans.

Bibliography

[1] M. Sun, A. Farhadi, and S. Seitz, "Ranking domain-specific highlights by analyzing edited videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 787–802.

[2] M. Gygli, Y. Song, and L. Cao, "Video2gif: Automatic generation of animated gifs from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1001–1009.

[3] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo, "Unsupervised extraction of video highlights via robust recurrent auto-encoders," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4633–4641.

[4] K. H. Zeng, T. H. Chen, J. C. Niebles, and M. Sun, "Generation for user generated videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 609–625.

[5] Y. Jiao, X. Yang, T. Zhang, S. Huang, and C. Xu, "Video highlight detection via deep ranking modeling," in *Proceedings of the Pacific-Rim Symposium on Image and Video Technology*, 2017, pp. 28–39.

[6] Y. Yu, S. Lee, J. Na, J. Kang, and G. Kim, "A deep ranking model for spatio-temporal highlight detection from a 360◦ video," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[7] A. Garcia del Molino and M. Gygli, "Phd-gifs: Personalized highlight detection for automatic gif creation," in *Proceedings of the 26th ACM International Conference on Multimedia*, 2018, pp. 600–608.

[8] B. Xiong, Y. Kalantidis, D. Ghadiyaram, and K. Grauman, "Less is more: Learning highlight detection from video duration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1258–1267.

[9] M. Rochan, M. K. K. Reddy, L. Ye, and Y. Wang, "Adaptive video highlight detection by learning from user history," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 261–278.

[10] F. T. Hong, X. Huang, W. H. Li, and W. S. Zheng, "Mini-net: Multiple instance ranking network for video highlight detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 345–360.

[11] L. Wang, D. Liu, R. Puri, and D. N. Metaxas, "Learning trailer moments in full-length movies," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 300–316.

[12] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4489–4497.

[13] J. Wang, C. Xu, E. Chng, and Q. Tian, "Sports highlight detection from keyword sequences using hmm," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, vol. 1, 2004, pp. 599–602.

[14] Z. Xiong, R. Radhakrishnan, A. Divakaran, and T. S. Huang, "Highlights extraction from sports video based on an audio-visual marker detection framework," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2005, pp. 29–32.

[15] H. Tang, V. Kwatra, M. E. Sargin, and U. Gargi, "Detecting highlights in sports videos: Cricket as a test case," in *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, 2011, pp. 1–6.

[16] T. Yao, T. Mei, and Y. Rui, "Highlight detection with pairwise deep ranking for first-person video summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 982–990.

[17] S. Cai, W. Zuo, L. S. Davis, and L. Zhang, "Weakly-supervised video summarization using variational encoder-decoder and web prior," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 184–200.

[18] W.-S. Chu, Y. Song, and A. Jaimes, "Video co-summarization: Video summarization by visual co-occurrence," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3584–3592.

[19] R. Panda and A. K. Roy-Chowdhury, "Collaborative summarization of topic-related videos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7083–7092.

[20] B. Mahasseni, M. Lam, and S. Todorovic, "Unsupervised video summarization with adversarial lstm networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 202–211.

[21] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury, "Weakly supervised summarization of web videos," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3657–3666.

[22] A. Khosla, R. Hamid, C. J. Lin, and N. Sundaresan, "Large-scale video summarization using web-image priors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2698–2705.

[23] G. Kim and E. P. Xing, "Reconstructing storyline graphs for image recommendation from web community photos," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 3882–3889.

[24] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 1346–1353.

[25] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2714–2721.

[26] C. W. Ngo, Y. F. Ma, and H. J. Zhang, "Automatic video summarization by graph modeling," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2003, pp. 104–109.

[27] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "Tvsum: Summarizing web videos using titles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 5179–5187.

[28] M. Rochan, L. Ye, and Y. Wang, "Video summarization using fully convolutional sequence networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 347–363.

[29] K. Zhou, Y. Qiao, and T. Xiang, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[30] K. Zhang, K. Grauman, and F. Sha, "Retrospective encoders for video summarization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 383–399.

[31] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid, "Category-specific video summarization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 540–555.

[32] M. Rochan and Y. Wang, "Video summarization by learning from unpaired data," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7902–7911.

[33] B. Gong, W. L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," *Proceedings of Advances in neural information processing systems (NeurIPS)*, vol. 27, pp. 2069–2077, 2014.

[34] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 505–520.

[35] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3090–3098.

[36] K. Zhang, W. L. Chao, F. Sha, and K. Grauman, "Summary transfer: Exemplar-based subset selection for video summarization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 1059–1067.

[37] K. Zhang, W. L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016, pp. 766–782.

[38] B. Zhao, X. Li, and X. Lu, "Hierarchical recurrent neural network for video summarization," in *Proceedings of the 25th ACM International Conference on Multimedia*, 2017, pp. 863–871.

[39] J. Fajtl, H. S. Sokeh, V. Argyriou, D. Monekosso, and P. Remagnino, "Summarizing videos with attention," in *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2018, pp. 39–54.

[40] R. Gao, T. H. Oh, K. Grauman, and L. Torresani, "Listen to look: Action recognition by previewing audio," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10 457–10 467.

[41] T. Afouras, J. S. Chung, A. Senior, O. Vinyals, and A. Zisserman, "Deep audio-visual speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2018.

[42] G. Sterpu, C. Saam, and N. Harte, "Attention-based audio-visual fusion for robust automatic speech recognition," in *Proceedings of the 20th ACM International Conference on Multimodal Interaction*, 2018, pp. 111–115.

[43] W. Wang, D. Tran, and M. Feiszli, "What makes training multi-modal classification networks hard?" In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 695–12 705.

[44] J. Lu, C. Xiong, D. Parikh, and R. Socher, "Knowing when to look: Adaptive attention via a visual sentinel for image captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 375–383.

[45] S. Becker and G. E. Hinton, "Self-organizing neural network that discovers surfaces in random-dot stereograms," *Nature*, vol. 355, no. 6356, pp. 161–163, 1992.

[46] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 2, 2006, pp. 1735–1742.

[47] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[48] T. Gao, X. Yao, and D. Chen, "Simcse: Simple contrastive learning of sentence embeddings," *arXiv preprint arXiv:2104.08821*, 2021.

[49] Q. Kong, Y. Cao, T. Iqbal, Y. Wang, W. Wang, and M. D. Plumbley, "Panns: Large-scale pretrained audio neural networks for audio pattern recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2880–2894, 2020.

[50] K. Hara, H. Kataoka, and Y. Satoh, "Can spatiotemporal 3d cnns retrace the history of 2d cnns and imagenet?" In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6546–6555.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[52] M. Yin, Z. Yao, Y. Cao, X. Li, Z. Zhang, S. Lin, and H. Hu, "Disentangled non-local neural networks," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020, pp. 191–207.

[53] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[55] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6299–6308.

[56] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 1725–1732.

[57] J. F. Gemmeke, D. P. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, "Audio set: An ontology and human-labeled dataset for audio events," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 776–780.

[58] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

[59] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," in *International Conference on Machine Learning (ICML)*, PMLR, 2020, pp. 1597–1607.

[60] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[61] T. Badamdorj, M. Rochan, Y. Wang, and L. Cheng, "Joint visual and audio learning for video highlight detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 8127–8137.

[62] Z. Chi, Y. Wang, Y. Yu, and J. Tang, "Test-time fast adaptation for dynamic scene deblurring via meta-auxiliary learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 9137–9146.

[63] D. Wang, E. Shelhamer, S. Liu, B. Olshausen, and T. Darrell, "Tent: Fully test-time adaptation by entropy minimization," *arXiv preprint arXiv:2006.10726*, 2020.

[64] C. Y. Wu, M. Zaheer, H. Hu, R Manmatha, A. J. Smola, and P. Krähenbühl, "Compressed video action recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6026–6035.