This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Contrastive Learning for Unsupervised Video Highlight Detection

Taivanbat Badamdorj[1] Mrigank Rochan[2] Yang Wang[2,3] Li Cheng[1]

[1]University of Alberta
[2]Huawei Noah's Ark Lab
[3]University of Manitoba

Abstract

Video highlight detection can greatly simplify video browsing, potentially paving the way for a wide range of applications. Existing efforts are mostly fully-supervised, requiring humans to manually identify and label the interesting moments (called highlights) in a video. Recent weakly supervised methods forgo the use of highlight annotations, but typically require extensive efforts in collecting external data such as web-crawled videos for model learning. This observation has inspired us to consider unsupervised highlight detection where neither frame-level nor video-level annotations are available in training. We propose a simple contrastive learning framework for unsupervised highlight detection. Our framework encodes a video into a vector representation by learning to pick video clips that help to distinguish it from other videos via a contrastive objective using dropout noise. This inherently allows our framework to identify video clips corresponding to highlight of the video. Extensive empirical evaluations on three highlight detection benchmarks demonstrate the superior performance of our approach.

1. Introduction

Video highlight detection aims to automatically find the interesting moments (called "highlights") within videos. With the explosion of video content in the past decade, it has become even more important due to its broad range of applications, such as video retrieval, recommendation, browsing, and editing. In recent years, there has been significant progress in video highlight detection. Existing efforts predominantly focus on the fully-supervised scenario [9, 14, 19, 31, 36, 42, 47, 48], which requires manual annotation of the highlight moments in the training videos. Since manual annotations are expensive to obtain, recent weakly supervised methods [3, 6, 18, 28, 29, 43] make use of video-level labels such as video category as a weak supervision signal. However, the state-of-the-art methods often rely on large-scale external data. For example, LM [43] employs 10 million Instagram videos to train their model.

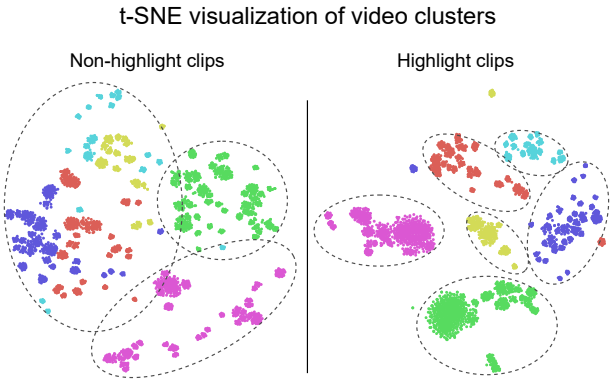

Figure 1. **Highlight detection using contrastive learning**. Our model identifies highlight clips in a video because they form better clusters for contrastive learning. We use t-SNE [39] to visualize a video and its nearest neighbors using non-highlight clips and highlight clips. Each point represents a video clip, and each color represents a video. We see that poor clusters in the non-highlight clip embedding space (left) are clearer in the highlight clip embedding space (right). Ellipses are drawn around the clusters to show that video clusters that are not separable in the non-highlight clip embedding space (red and dark blue, yellow and light blue) are separated in the highlight clip embedding space. The green, yellow, and pink videos also form tighter clusters. Thus, our model learns to pick highlight clips to do well on the contrastive learning task. See text for detailed explanation.

Recent advances in contrastive learning have helped close the gap between unsupervised models and their supervised counterparts [5, 8]. 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 this domain, a model is trained to map

Code: https://github.com/tkbadamdorj/CHD.

two randomly transformed versions of the same image (*e.g.* random cropping, color distortion) close together in an embedding space, while mapping the same image farther from other images that are also randomly transformed [5]. Thus, we can view contrastive learning as a clustering task: we want 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. Recent work [8] has found that dropout [35] can be used as a random transformation to learn superior unsupervised sentence embeddings through contrastive learning. Dropout is a general transformation that can be applied to any network.

Inspired by this idea, we propose a novel and simple unsupervised[1] framework for video highlight detection. In the highlight detection task, we break down each video into fixed length clips (*e.g.* 100 frames). Then we use a pretrained feature extractor such as a C3D action recognition model [38] to obtain a vector representation of each clip. Thus, we represent an input video by a sequence of vectors. We interchangeably use "clip" to refer to the vector representation of a clip. Our unsupervised framework picks clips to produce a single vector embedding for the entire video. 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 close together, while mapping the video farther from other videos.

Our model learns to pick highlight clips (interesting moments) since they contain more information about the video content. Let us motivate this claim through the example in Fig. 1. We learn to cluster a video close to itself under random dropout perturbations. We claim that highlight clips contain more information about the video itself, thus better clusters are formed if our model picks highlight clips to produce a video embedding. In this figure, we show the same set of videos represented purely by non-highlight clips (left) and highlight clips (right) using t-SNE [39]. Each video is shown using a different color, and each point represents a highlight/non-highlight clip after we apply a series of random dropouts within the network. 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 experimentally prove this claim in Section 4.3.

[1]We use *unsupervised* to indicate that the method does not have access to any manually annotated training data or video-level labels. Weakly supervised methods typically utilize video-level label (category) or video length information (short web videos) for supervision.

In short, our main contribution is a novel unsupervised framework for video highlight detection based on contrastive learning. Empirical evaluation on three widely-used highlight detection benchmarks demonstrate the superior performance of our approach. In many cases, it performs on par or better than the state-of-the-art methods that make use of large amounts of external data.

2. Related Work

Video Highlight Detection: Early efforts in video highlight detection mainly deal with sports videos [37, 41, 44]. Later works were proposed to tackle a broader range of applications including social media [36] and first-person videos [46].

Most methods typically consider a fully-supervised scenario that requires dense frame-level annotations [9, 14, 19, 31, 36, 42, 47]. Weakly-supervised highlight detection has been considered by several recent works [3,6,18,28,29,43], where the training label is only available at the video-level. They typically require access to large-scale external datasets. For example, the work of [43] takes advantage of the fact that clips from shorter videos are more likely to be highlights to train a ranking network. MN [18], 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.

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 [21, 22, 24–27, 29, 33, 34, 51, 53], and as such, many rely on heuristics such as diversity and representativeness to obtain a summary video. Weakly supervised methods [3,21,22,28,30,32,34] have also been developed to utilize video-level annotations. Benefiting from the massive user tagging of online videos, research efforts in supervised learning [7,10,12,13,33,49,50,52] are also progressing rapidly. Our work is influenced by the attention-based model of [7] that operates on sequences of clips as potential highlights.

Contrastive Learning: The idea of contrastive learning is to make representations of a sample agree under small transformations [2]. 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 a representation of the same image with different random transformations [5]. 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 useful representation for downstream tasks such as image classification.

In other words, we aim to learn effective representations by learning to pull semantically close neighbors together

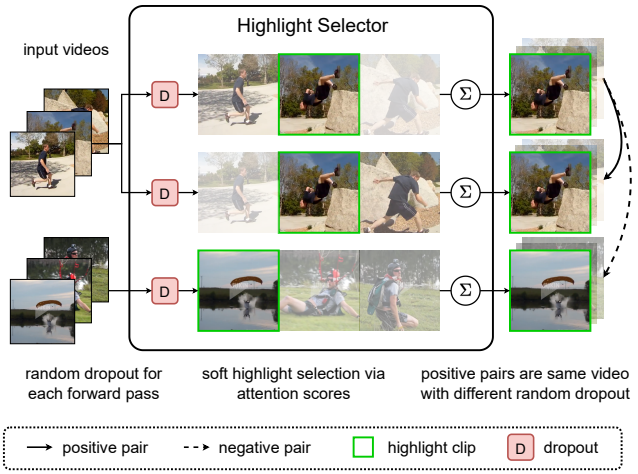

Figure 2. **Unsupervised learning framework.** The highlight selector picks clips that help increase the similarity between a positive video pair, while decreasing the similarity between negative video pairs. We construct a positive video pair by applying two different hidden dropout masks to the same input video. We form negative pairs by sampling other videos. We show that our model picks highlight clips because they form better clusters for contrastive learning in Sec. 4.3.

while pushing apart non-neighbors [15]. Recent work has suggested using dropout [35] within the model as a form of data augmentation to learn unsupervised sentence embeddings [8]. This allows us to obtain different representations for the same input by using different random dropout masks. We use dropout to obtain different representations of a video in our highlight selector model. We refer the reader to the SimCLR paper [5] for more related efforts in contrastive learning.

3. Our Approach

We split a video into fixed length clips and get N resulting clips. We represent each clip using a vector $\boldsymbol{v}_i \in \mathbb{R}^d$ and represent the video using the sequence of clips $\boldsymbol{V} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_N\}$. We extract the raw features of dimension d_v with a pre-trained visual feature extractor, such as a C3D action recognition model [38], which we then linearly projected to size d to obtain the features $\boldsymbol{v}_i$.

Our goal is then to produce highlight scores $S = \{s_1, \ldots, s_N\}$ where s_i is the score of clip $\boldsymbol{v}_i$. We achieve this through our contrastive learning framework shown in Fig. 2, where we learn to map a transformed version of a sample close to itself and farther away from other samples in an embedding space. For this, we use different dropout masks in our network to produce a transformed version of each video. The clip selector generates an attention score α_j for each clip $\boldsymbol{v}_j$ to compute a weighted sum of the clips. We use this weighted sum 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. We use the attention scores $\alpha = \{\alpha_1, \ldots, \alpha_N\}$ from the clip selector directly as our highlight scores $S = \{s_1, \ldots, s_N\}$.

3.1. 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 [40] as illustrated in Fig. 3. 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 convenience:

$$\alpha, \boldsymbol{Z} = \text{Attn}(\boldsymbol{X}, \boldsymbol{Y}). \tag{1}$$

The details of Eq. (1) are as follows. First, we define the query, key, and value linear transformations:

$$\boldsymbol{q}_i = \boldsymbol{W}_q \boldsymbol{x}_i, \; \boldsymbol{k}_j = \boldsymbol{W}_k \boldsymbol{y}_j, \; \boldsymbol{v}_j = \boldsymbol{W}_v \boldsymbol{y}_j, \tag{2}$$

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.

$$\alpha_{ij} = \text{softmax}(\boldsymbol{q}_i \cdot \boldsymbol{k}_j). \tag{3}$$

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

$$\hat{\boldsymbol{z}}_i = \sum_{j=1}^{N} \alpha_{ij} \boldsymbol{v}_j. \tag{4}$$

We then use a feed-forward network sandwiched by layer normalization layers on either side. The feed-forward network is defined as $\text{FF}(\boldsymbol{x}_i) = \max(0, \boldsymbol{x}_i \boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2$. $\boldsymbol{W}_1, \boldsymbol{W}_2 \in \mathbb{R}^{d \times d}$, $\boldsymbol{b}_1, \boldsymbol{b}_2 \in \mathbb{R}^d$ are the weights and biases, respectively, of the feed-forward network. In addition, we include residual connections before the layer normalization layers as shown in Fig. 3. Thus, we obtain our final attended features $\boldsymbol{z}_i$.

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 cannot use the residual connections and remove them from the network.

3.2. Highlight Selector

We show our highlight selector architecture in Fig. 4. The highlight selector takes in N fixed-length clips that represent the content of the video, and outputs a single vector

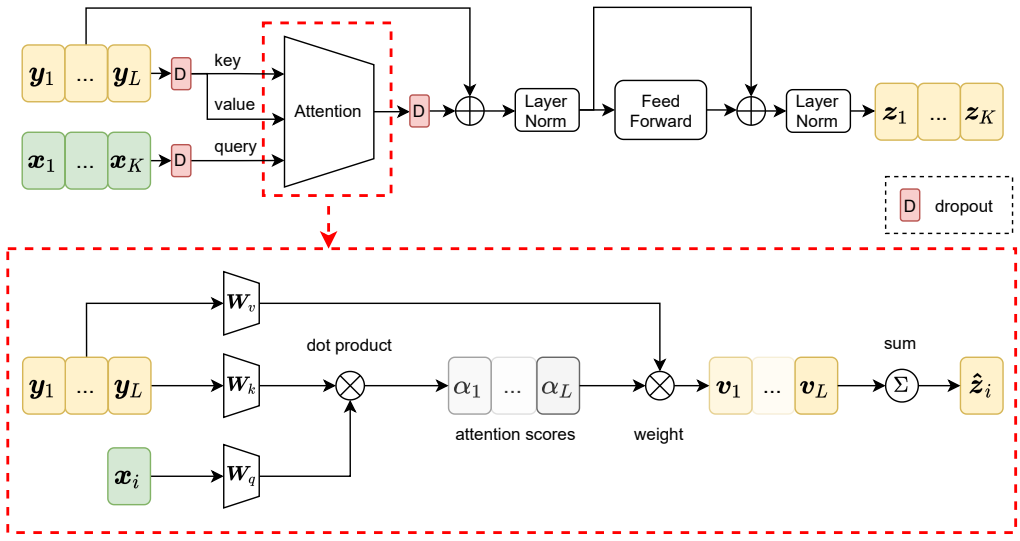

Figure 3. **Generic attention layer.** The generic attention layer takes in a query sequence $\boldsymbol{x}_i,\ i = 1, ..., K$, and a key/value sequence $\boldsymbol{y}_j,\ j = 1, ..., L$. It then produces an attended feature $\boldsymbol{z}_i$ for each query $\boldsymbol{x}_i$. We apply dropout to 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. We use residual connections when $L = K$.

embedding $\boldsymbol{h}$ that represents the entire video. It achieves this by learning to produce an attention score α_j for each clip $\boldsymbol{v}_j$ that decides the importance of clip $\boldsymbol{v}_j$ in representing the video. We use the attention score α_j directly as our highlight score s_j.

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 first use an attention layer. First, we denote the raw video features obtained from a pre-trained feature extractor as $\hat{\boldsymbol{v}}_i \in \mathbb{R}^{d_v}$, where d_v is the size of the vector. We linearly project these features to size d:

$$\boldsymbol{v}_i = \boldsymbol{W}_{\mathcal{V}} \hat{\boldsymbol{v}}_i, \tag{5}$$

where $\boldsymbol{W}_{\mathcal{V}} \in \mathbb{R}^{d \times d_v}$. We use $\boldsymbol{V}$ to denote the set of features of a video, i.e. $\boldsymbol{V} = \{\boldsymbol{v}_1, \boldsymbol{v}_2, ..., \boldsymbol{v}_N\}$. Then we use a generic attention layer to perform self-attention:

$$\boldsymbol{Z}^v = \{\boldsymbol{z}_1^v, \boldsymbol{z}_2^v, ..., \boldsymbol{z}_N^v\} = \text{Attn}(\boldsymbol{V}, \boldsymbol{V}). \tag{6}$$

Note that the self-attention is a specific case of our generic attention layer from Sec. 3.1, where the sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ are identical.

We use another attention layer to directly pick highlight clips. 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 as the output of our attention layer since the number of queries K is 1:

$$\bar{\boldsymbol{v}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{z}_i^v, \tag{7}$$

$$\alpha, \{\boldsymbol{h}\} = \text{Attn}(\{\bar{\boldsymbol{v}}\}, \boldsymbol{Z}^v). \tag{8}$$

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 highlight selector as $\boldsymbol{h} = F(\boldsymbol{V})$, and use the attention score α_j directly as our highlight score s_j for clip $\boldsymbol{v}_j$.

3.3. Contrastive Learning

We illustrate our contrastive learning framework in Fig. 2. We learn to map a transformed version of a sample close to itself relative to other samples in an embedding space. In this regard, we adopt the SimCSE framework, and use dropout as our transformation [8] to facilitate different embeddings of the same video input.

Dropout: In our model, we apply dropout in two places for each generic attention layer. First, we apply it to the inputs. Then we apply it to the output of the dot-product attention $\hat{\boldsymbol{z}}_i$ before the layer-normalization and feed-forward layer. We show this in Fig. 3.

We use $\boldsymbol{m}$ to denote the entire sequence of dropout masks applied within the network for an input video. Given

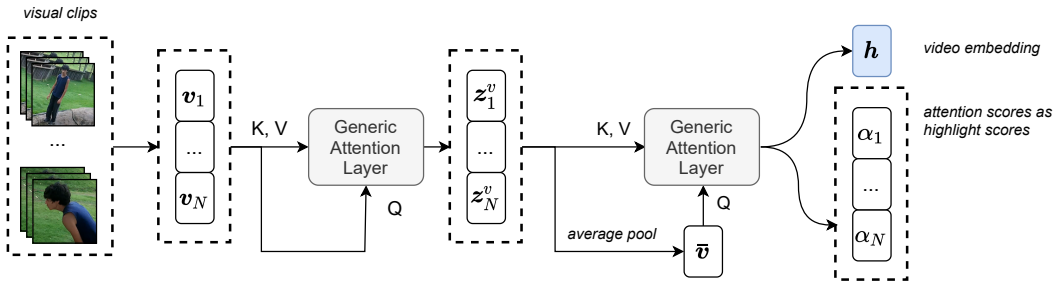

Figure 4. **Highlight selector architecture.** We break a video into clips of some fixed number of frames (*e.g.* 100 frames) and obtain N clips. For each clip we obtain features from a pre-trained feature extractor and project it to to size d to get $\boldsymbol{v}_j$. We use a generic attention layer to model the relationship between the clips. We then use another generic attention layer using the average-pooled features as the query $\bar{\boldsymbol{v}}$. This produces a final set of attention scores α_j and a video embedding $\boldsymbol{h}$. We use the attention scores α_j directly as our highlight scores s_j. These scores are indirectly unsupervised through the video embedding $\boldsymbol{h}$ which we use for the contrastive learning task.

this mask, we denote the embedding produced by our highlight selector as $\boldsymbol{h}^{\boldsymbol{m}}$.

Contrastive Loss: We adopt the common softmax formulation of the contrastive loss [5, 8]. For a video l, we apply two different dropout masks $\boldsymbol{m}_l$ and $\boldsymbol{m}'_l$ to produce the positive pair $\{\boldsymbol{h}_l^{m_l}, \boldsymbol{h}_l^{m'_l}\}$. We randomly sample other videos as negative samples. Our task is then to increase the similarity between the positive pair while decreasing the similarity between negative pairs. Concretely, we formulate the loss for video l as follows:

$$\mathcal{L}_l = -\log \frac{\exp(\text{sim}(\boldsymbol{h}_l^{m_l}, \boldsymbol{h}_l^{m'_l})/\tau)}{\sum_{k=1}^{B} \exp(\text{sim}(\boldsymbol{h}_l^{m_l}, \boldsymbol{h}_k^{m'_k})/\tau)}. \quad (9)$$

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

4. Experiments

4.1. Datasets and Setup

We carry out empirical evaluations on three benchmark datasets, namely the YouTube Highlights dataset [36], TVSum dataset [34], as well as the Video Titles in the Wild (VTW) dataset [48]. The YouTube Highlights dataset [36] contains videos of six different categories: dog, gymnastics, parkour, skating, skiing, and surfing. We pool all the videos together, and train a general model for all categories. The YouTube dataset comes with a canonical training and test split. 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 [28, 29]. The Video Titles in the Wild (VTW) dataset [48] 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 [48] and adopt their split of 2,000 videos for training, and 2,000 for testing.

Features: On the YouTube Highlights and TVSum datasets, we follow prior works [42, 43] and use a 3D CNN [16] with ResNet-34 [17] backbone pre-trained on the Kinetics-400 dataset [4] 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. We average pool the frame-level features to obtain a clip-level feature.

On the VTW dataset, we follow the original work of [48] and use a C3D network [38] pre-trained on the Sports-1M [20] dataset to obtain visual features. We divide each video into clips of 100 frames following the original work [48].

Implementation Details: We train our model using the Adam optimizer [23] 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 B 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. We initially started with the code-base from the work of [7][2], but ended up heavily modifying their code for our purposes.

Evaluation Metrics: We use mean Average Precision (mAP) to evaluate our model. Following prior studies [14, 36, 42], 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. On the TVSum dataset, we report the mean Average Precision on the top 5 predicted clips (top-5 mAP).

Comparison Methods: We compare our method to several state-of-the art methods on the YouTube dataset: RRAE [45], GIFs [14], LSVM [36], LM-A, LM-S [43], MN [18],

[2] https://github.com/ok1zjf/VASNet

Trailers [42], and Attn [1]. Prior weakly supervised methods RRAE, LM-A, LM-S, and MN rely on the collection of large external datasets, in contrast to our model.

On the TVSum dataset, we compare our method to prior unsupervised and weakly supervised methods: MBF [6], CVS [29], SG [26], VESD [3], LM-S [43], MN [18], LM-A [43], and DSN [28]. Of these works VESD, LM-A, LM-S, and MN make use of external data.

RRAE and VESD collect edited web videos (videos that have been edited to only contain highlights) to weakly supervise their network. RRAE trains an auto-encoder on edited videos. Their intuition is that non-highlight clips from unedited videos will have a higher reconstruction error at test-time. VESD uses edited web videos as a prior and aim to make the distribution of the output highlights match the distribution of the web videos. LM-A and LM-S collect 10 million videos from Instagram. They posit that clips from short videos are more likely to be highlight clips since they are more likely to have been edited by users to preserve the main content of the video. Then they train a ranking network to rank clips from short videos higher than clips from long videos. MN collects unedited videos with category-level information; they use category-level information as their weak supervision signal.

MBF and CVS don't use external data, but use videos from the same category to collaboratively detect highlights. DSN also uses category-level information as a weak supervision signal. They train a classification network that predicts the video category. They then compute the highlight scores for each clip via back-propagation guided by the category with the highest score in the forward pass. SG is the only other work that is entirely unsupervised. They train a generative adversarial network [11] to pick frames that preserve the original video content while being different from a network that outputs the same highlight score for all clips.

While we have fewer baselines on the VTW dataset, we compare our unsupervised model to prior supervised works [1, 48] and show competitive performance.

4.2. Unsupervised Highlight Detection

YouTube: We compare our model to other weakly supervised methods in Table 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 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 2. We outperform two prior supervised works, and achieve a performance only 5.1% below the supervised state-of-the-art, Attn [1]. In addition, we achieve comparable performance to Attn across most categories.

Method	RRAE [45]	LM-A [43]	LM-S [43]	MN [18]	Ours
Ext. 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 1. **Highlight detection results on the YouTube dataset**. We compare (in terms of mAP) with weakly supervised methods. Our model outperforms all prior methods by a large margin. To our knowledge, no prior unsupervised or weakly supervised work exists that does not make use of external data on the YouTube dataset.

Method	GIFs [14]	LSVM [36]	Trail. [42]	Attn. [1]	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 2. **Comparison with supervised methods on the YouTube dataset.** Our model achieves competitive performance compared to state-of-the-art supervised highlight detection methods, outperforming two prior supervised works, and achieving a mAP only 5.1% below the supervised state-of-the-art model Attn.

TVSum: We compare our model to other unsupervised and weakly supervised models on the TVSum dataset that don't use external data in Table 3. Our model outperforms all prior methods. In particular, we outperform the state-of-the-art significantly by 6.5%.

We also compare our method to weakly supervised models that make use of external data in Table 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.

Method	MBF [6]	CVS [29]	SG [26]	DSN [28]	Ours
Ext. data	✗	✗	✗	✗	✗
mAP	0.345	0.372	0.462	0.424	**0.5276**

Table 3. **Highlight detection results on the TVSum dataset.** We compare (top-5 mAP) with unsupervised and weakly supervised methods that *don't use* external data. Our method outperforms the prior methods by a large margin.

Method	VESD [3]	LM-A [43]	LM-S [43]	MN [18]	Ours
Ext. data	✓	✓	✓	✓	✗
mAP	0.423	0.524	0.563	0.6979	0.5276

Table 4. **Highlight detection results on the TVSum dataset**. We compare (top-5 mAP) with weakly supervised methods that *use* external data. Even without external data our model outperforms two prior methods.

VTW: We present our unsupervised learning results for VTW in Table 5. 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 [50], while maintaining reasonable performance relative to the supervised state-of-the-art, Attn [1]. We outperform the prior supervised work by 2.6%, and underperform relative to the supervised 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.

Method	VTW [48]	Attn. [1]	Ours
Supervised	✓	✓	✗
mAP	0.583	0.722	0.6090

Table 5. **Highlight detection results on the VTW dataset**. We achieve comparable performance (mAP) with a fully supervised state-of-the-art methods and outperform the other supervised method.

4.3. 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. 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 $\boldsymbol{z}_i^v$ at the output of the first generic attention layer.

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 denote the i-th clip of video k by $\boldsymbol{h}_{ik}$. The centroid is simply the average of all the sampled clip embeddings from video k: $\bar{\boldsymbol{h}}_k = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{h}_{ik}$. 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.

The inter-cluster distance is a measure of 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 across all videos.

We show our results in Table 6. We can see that non-highlight clips form poorer clusters. Highlight clips form a tighter cluster in the embedding space: the intra-cluster distance is high for non-highlight clips, and low for highlight clips. In addition, highlight clips form clusters that are well separated: the inter-cluster distance is high for using highlight clips and low for non-highlight clips. 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.

	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 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.

4.4. Effect of Dropout on Performance

We investigate the effect of dropout in this section, and show the results in Table. 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 positive pair would simply be 1 because they are identical. However, our model would still have to learn to decrease the similarity between negative pairs. In consequence, this boils down to a repelling loss where our model tries to map different videos as far away as possible in the embedding space. As such, our results are still competitive to prior works for all three datasets.

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

Table 7. **Effect of dropout on performance**. We use a dropout rate p of 0.1 for all three datasets.

4.5. Qualitative Results

We show qualitative results in Fig. 5 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.

Figure 5. **Qualitative results**. We show some qualitative examples. Ground truth highlights are marked in green. In the skateboarding video (top), there are two highlight segments. While our model chooses to focus mostly on the second highlight segment, we see that it detects the first as well. In the second video of somebody crashing while paragliding, our model correctly identifies the highlight segment.

5. Conclusion and Limitations

In this paper, 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.

However, our unsupervised framework still has several limitations. We have found empirically that non-highlight clips are not as informative about the video on the three datasets that we have used, but there is no guarantee that this is always true. It could be the case that some videos have sufficiently different content that there is no need to select highlight clips to differentiate between different videos. This could however be tackled through negative hard-mining i.e. by picking videos that are closest to a given video to form negative pairs. In this paper, we have simply used random sampling to pick the videos.

Another limitation is in how we can best incorporate available labeled data. It is usually the case that we have at least some videos with the highlight clips labeled, so it would be interesting to tackle a semi-supervised framework that also incorporates labeled examples.

References

[1] Taivanbat Badamdorj, Mrigank Rochan, Yang Wang, and Li Cheng. Joint visual and audio learning for video highlight detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 8127–8137, 2021. 6, 7

[2] Suzanna Becker and Geoffrey E Hinton. Self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature*, 355(6356):161–163, 1992. 2

[3] Sijia Cai, Wangmeng Zuo, Larry S Davis, and Lei Zhang. Weakly-supervised video summarization using variational encoder-decoder and web prior. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 184–200, 2018. 1, 2, 6, 7

[4] Joao Carreira and Andrew 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)*, pages 6299–6308, 2017. 5

[5] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pages 1597–1607. PMLR, 2020. 1, 2, 3, 5

[6] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. Video co-summarization: Video summarization by visual co-occurrence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3584–3592, 2015. 1, 2, 6, 7

[7] Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso, and Paolo Remagnino. Summarizing videos with attention. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, pages 39–54, 2018. 2, 5

[8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*, 2021. 1, 2, 3, 4, 5

[9] Ana Garcia del Molino and Michael Gygli. Phd-gifs: personalized highlight detection for automatic gif creation. In *Proceedings of the 26th ACM International Conference on Multimedia*, pages 600–608, 2018. 1, 2

[10] Boqing Gong, Wei Lun Chao, Kristen Grauman, and Fei Sha. Diverse sequential subset selection for supervised video summarization. *Proceedings of Advances in neural information processing systems (NeurIPS)*, 27:2069–2077, 2014. 2

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 6

[12] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool. Creating summaries from user videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 505–520, 2014. 2

[13] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3090–3098, 2015. 2

[14] Michael Gygli, Yale Song, and Liangliang Cao. Video2gif: Automatic generation of animated gifs from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1001–1009, 2016. 1, 2, 5, 6

[15] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 1735–1742. IEEE, 2006. 3

[16] Kensho Hara, Hirokatsu Kataoka, and Yutaka 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)*, pages 6546–6555, 2018. 5

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 5

[18] Fa Ting Hong, Xuanteng Huang, Wei Hong Li, and Wei Shi Zheng. Mini-net: Multiple instance ranking network for video highlight detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 345–360, 2020. 1, 2, 5, 6, 7

[19] Yifan Jiao, Xiaoshan Yang, Tianzhu Zhang, Shucheng Huang, and Changsheng Xu. Video highlight detection via deep ranking modeling. In *Proceedings of the Pacific-Rim Symposium on Image and Video Technology*, pages 28–39, 2017. 1, 2

[20] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1725–1732, 2014. 5

[21] Aditya Khosla, Raffay Hamid, Chih Jen Lin, and Neel Sundaresan. Large-scale video summarization using web-image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2698–2705, 2013. 2

[22] Gunhee Kim and Eric 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)*, pages 3882–3889, 2014. 2

[23] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014. 5

[24] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1346–1353, 2012. 2

[25] Zheng Lu and Kristen Grauman. Story-driven summarization for egocentric video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2714–2721, 2013. 2

[26] Behrooz Mahasseni, Michael Lam, and Sinisa Todorovic. Unsupervised video summarization with adversarial lstm

networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pages 202–211, 2017. 2, 6, 7

[27] Chong Wah Ngo, Yu Fei Ma, and Hong Jiang Zhang. Automatic video summarization by graph modeling. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 104–109, 2003. 2

[28] Rameswar Panda, Abir Das, Ziyan Wu, Jan Ernst, and Amit K Roy-Chowdhury. Weakly supervised summarization of web videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3657–3666, 2017. 1, 2, 5, 6, 7

[29] Rameswar Panda and Amit K Roy-Chowdhury. Collaborative summarization of topic-related videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7083–7092, 2017. 1, 2, 5, 6, 7

[30] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 540–555, 2014. 2

[31] Mrigank Rochan, Mahesh Kumar Krishna Reddy, Linwei Ye, and Yang Wang. Adaptive video highlight detection by learning from user history. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 261–278, 2020. 1, 2

[32] Mrigank Rochan and Yang Wang. Video summarization by learning from unpaired data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7902–7911, 2019. 2

[33] Mrigank Rochan, Linwei Ye, and Yang Wang. Video summarization using fully convolutional sequence networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 347–363, 2018. 2

[34] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes. Tvsum: Summarizing web videos using titles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5179–5187, 2015. 2, 5

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. 2, 3

[36] Min Sun, Ali Farhadi, and Steve Seitz. Ranking domain-specific highlights by analyzing edited videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 787–802, 2014. 1, 2, 5, 6

[37] Hao Tang, Vivek Kwatra, Mehmet Emre Sargin, and Ullas Gargi. Detecting highlights in sports videos: Cricket as a test case. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6, 2011. 2

[38] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4489–4497, 2015. 2, 3, 5

[39] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 1, 2

[40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2017. 3

[41] Jinjun Wang, Changsheng Xu, Engsiong Chng, and Qi Tian. Sports highlight detection from keyword sequences using hmm. In *Proceedings of the IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, pages 599–602, 2004. 2

[42] Lezi Wang, Dong Liu, Rohit Puri, and Dimitris N Metaxas. Learning trailer moments in full-length movies. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 300–316, 2020. 1, 2, 5, 6

[43] Bo Xiong, Yannis Kalantidis, Deepti Ghadiyaram, and Kristen Grauman. Less is more: Learning highlight detection from video duration. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1258–1267, 2019. 1, 2, 5, 6, 7

[44] Ziyou Xiong, Regunathan Radhakrishnan, Ajay Divakaran, and Thomas 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)*, pages 29–32, 2005. 2

[45] Huan Yang, Baoyuan Wang, Stephen Lin, David Wipf, Minyi Guo, and Baining Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 4633–4641, 2015. 5, 6

[46] Ting Yao, Tao Mei, and Yong 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)*, pages 982–990, 2016. 2

[47] Youngjae Yu, Sangho Lee, Joonil Na, Jaeyun Kang, and Gunhee Kim. A deep ranking model for spatio-temporal highlight detection from a 360◦ video. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 1, 2

[48] Kuo Hao Zeng, Tseng Hung Chen, Juan Carlos Niebles, and Min Sun. Generation for user generated videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 609–625, 2016. 1, 5, 6, 7

[49] Ke Zhang, Wei Lun Chao, Fei Sha, and Kristen Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1059–1067, 2016. 2

[50] Ke Zhang, Wei Lun Chao, Fei Sha, and Kristen Grauman. Video summarization with long short-term memory. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 766–782, 2016. 2, 7

[51] Ke Zhang, Kristen Grauman, and Fei Sha. Retrospective encoders for video summarization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 383–399, 2018. 2

[52] Bin Zhao, Xuelong Li, and Xiaoqiang Lu. Hierarchical recurrent neural network for video summarization. In *Proceedings of the 25th ACM International Conference on Multimedia*, pages 863–871, 2017. 2

[53] Kaiyang Zhou, Yu Qiao, and Tao Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2