Efficient Deep Learning Models for Video Abstraction

by

Mrigank Rochan

A Thesis submitted to the Faculty of Graduate Studies of
The University of Manitoba
in partial fulfilment of the requirements of the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science
University of Manitoba
Winnipeg

Thesis advisor

Yang Wang

Author

Mrigank Rochan

Efficient Deep Learning Models for Video Abstraction

Abstract

With the revolution in digital video technology, video data are ubiquitous and explosively growing. There is a compelling need to develop efficient automated techniques to manage video data. Therefore, *video abstraction* is of significant interest to the computer vision research community. The objective in video abstraction is to automatically create a short visual summary of a long input video so that a user can get certain perspectives of the video without watching or accessing it entirely. This mechanism would allow to easily preview, categorize, search, and edit the huge amount of video data. In this thesis, we push the state of the art in video abstraction in several ways. Firstly, we develop fully convolutional sequence deep learning models that address the computational limitations of the previous deep learning models for video abstraction. Secondly, we propose a new formulation of unpaired training data for the model learning to reduce the need of expensive labeled training data for supervised learning. Thirdly, since video abstraction has a degree of subjectiveness to it, we realize a model that yields personalized and user-specific predictions by referring to the user's previously created summaries. Lastly, we extend this user adaptive model such that it can handle natural language textual queries from users and make predictions that are semantically related to the queries. Although we focus on video

abstraction in this thesis, we believe that our models can potentially be applied to other video understanding problems (e.g., video classification, action recognition, and video captioning).

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my advisor Prof. Yang Wang for his support and guidance without which this thesis would not have been possible. His enthusiasm and dedication towards research has consistently motivated me. I am thankful to him for his timely and insightful feedback that played a crucial role in successfully completing the research in this thesis. I am grateful to him for creating time on weekdays, weekends and even holidays, for research discussions. I must also thank him for teaching me different aspects of research in computer vision and the craft of effectively communicating ideas.

I gratefully acknowledge the positive and encouraging feedback from my thesis committee members Prof. Jim Little, Prof. Lorenzo Livi, and Prof. Ekram Hossain. I thank them for serving on my committee.

Besides the generous funding I received from my advisor, I acknowledge the funding from the University of Manitoba Graduate Fellowship (UMGF) by the Faculty of Graduate Studies and the Guaranteed Funding Package (GFP) by the Department of Computer Science. I would also like to thank the support staff in the department, especially Lynne Hermiston and Gilbert Detillieux, for their help.

I am fortunate to have wonderful friends like Devinder, Nikhil, Peter, Hao, Kyle, Terry, Aoran, Debajyoti, Saeed, Praveen, Arjun, and Shyam, who made my life enriching and enjoyable. I am also extremely thankful to all my labmates for the amazing collaboration and interesting discussions.

Finally, I am deeply thankful to my family for their unwavering faith and encouragement. I cannot imagine reaching this far without their support and care.

This thesis is dedicated to my parents.

Publications

Most of the materials, ideas and figures in this thesis have previously appeared in the following publications by the author:

1. **M. Rochan**, L. Ye, and Y. Wang, "Video Summarization Using Fully Convolutional Sequence Networks," *European Conference on Computer Vision (ECCV)*, 2018, Springer Nature.

2. **M. Rochan** and Y. Wang, "Video Summarization by Learning From Unpaired Data," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

3. **M. Rochan**, M. K. Krishna Reddy, L. Ye, and Y. Wang, "Adaptive Video Highlight Detection by Learning from User History," *European Conference on Computer Vision (ECCV)*, 2020, Springer Nature (accepted and forthcoming).

4. **M. Rochan**, M. K. Krishna Reddy, and Y. Wang, "Sentence Guided Temporal Modulation for Dynamic Video Thumbnail Generation," *British Machine Vision Conference (BMVC)*, 2020 (accepted and forthcoming).

Chapter 1

Introduction

With the ever-increasing popularity and decreasing cost of video capture devices, the amount of video data has increased drastically in the past few years. According to Cisco Visual Networking Index 2017 [1], it is estimated that it will take around 5 million years for an individual to watch all the videos that are uploaded on the Internet each month in 2021! Video has become a prevalent mode of visual data. On the one hand, the huge video data present an exciting opportunity to make major scientific advances in video understanding applications such as search, recommendation, browsing, security, robotics, and video analytics. But on the other hand, it is challenging to exploit and manage this enormous amount of available video data for many of these applications. It is unrealistic for humans to browse and watch these videos to identify relevant and useful information. Hence, it is becoming increasingly important to develop automated methods that enable easier ways to preview, search, edit, and categorize videos. Consequently, *video abstraction* has emerged as an exciting area of research in computer vision. Video abstraction is an automatic

technique for generating a short and compact visual summary of a long original (raw or unedited) video [35; 84].

Based on the perspective of a video that needs to be retained in its summary, video abstraction can take various forms that are suitable for different applications [84]. In this thesis, we focus on three forms of video abstraction, namely video summarization, video highlight detection, and dynamic video thumbnail generation. In video summarization, the goal is to produce a short overview of a video (e.g., documentaries and instructional videos) [35; 48; 84] while satisfying certain properties such as diversity (no redundancy), interestingness (capture important objects crucial in visual narration), and representativeness (describe the main content of the video) [23]. In video highlight detection, the goal is to reduce a video to its interesting and important moments/events (e.g., movie trailers and sports highlights) [35; 48; 84]. Recently, a new task of dynamic video thumbnail generation is proposed which aims to compose a video thumbnail by selecting and concatenating a small number of clips of a video that may not be consecutive but are semantically related to the user query sentence [99]. The idea is to dynamically generate video thumbnails that match with user search intentions in video search websites like YouTube.

The output from video abstraction is commonly represented as either *keyframes* or *interval-based keyshots*. Keyframes [14; 23; 60; 52; 93; 103] are a subset of isolated defining frames of the video. Interval-based keyshots [26; 27; 68; 79; 103] are a set of segments or subshots of the video, where each segment or subshot consists of temporally continuous frames extending for a small time duration.

Video abstraction is a promising tool and mechanism to cope with the overwhelm-

ing amount of video data. It can also be useful in many real world applications. For example, in video surveillance, it is tedious and time-consuming for humans to browse through many hours of videos captured by surveillance cameras. If we can provide a short summary video that captures the important and meaningful information from a long video, it will greatly reduce human efforts required in video surveillance. Video abstraction can also provide better user experience in video search, retrieval, editing and understanding. Since short videos are easier to store and transfer, they can be effective in mobile applications. Furthermore, the summary videos can help in many downstream video analysis tasks, for instance, it is faster to run any other analysis algorithms (e.g., action recognition) on short videos. Thus, it is critical to develop models that can automatically understand the content of a video and produce a meaningful and optimal shorter version of it.

Despite being previously studied in the computer vision literature, the problem of video abstraction is unsolved and the state-of-the-art techniques (which are mainly deep learning based) are prone to various technical limitations. One limitation is that prior best methods mainly use recurrent deep learning models (e.g., Long Short-Term Memory (LSTM) [30]) in their approach. Since temporal information (such as motion, actions) in the frames of a video is helpful to perform video abstraction, recurrent models are natural and popular choice. This is due to their design and operational ability which is suitable to learn the temporal relationships in sequential data, for instance, videos. However, these models process the input data sequentially, e.g., for an input video it will process one frame at a time from left (first frame) to right (last frame). As a result, these models are hard to parallelize so as to take full advantage of

modern graphics processing units (GPUs) hardware. Another limitation is that prior superior methods are mainly supervised that require human-labeled training data for model learning which is very challenging and expensive to collect. Additionally, applications of many existing methods are also limited due to the fact that they usually learn a generic model for video abstraction, whereas the notion of a summary or highlight of a video is to a degree subjective [16]. Different users may not share same interests, and as a result it is very unlikely that different users will create exactly same summary for a given video [16]. Even though there is a degree of subjectivity in video abstraction, the major thrust of research is in developing generic models.

In this thesis, we conduct research to tackle the above limitations. We propose efficient deep learning models for video abstraction and advance the state of the art in video abstraction in several ways. Firstly, we develop non-recurrent deep learning models that are computational efficient than the existing deep learning based methods. Secondly, we introduce a new learning formulation that minimizes the need of labeled training data. Thirdly, we incorporate user preference (available in the form of past video highlights created by the user) in our learning framework and propose an adaptive model that is capable to make personalized and user-specific predictions. Lastly, we extend our model to include and leverage natural language textual queries from users while generating a concise preview of a video.

We envision that the technical and experimental contributions of this thesis will not only stimulate further research in video abstraction but also has potential to influence a variety of problems in video understanding (e.g., action recognition, moment localization, and video caption generation) and many application domains (e.g.,

healthcare and surveillance).

1.1 Thesis Contributions

In this thesis, we develop efficient deep learning models for video abstraction. In particular, we make the following contributions:

- We propose fully convolutional sequence models for video summarization (Chapter 3). Unlike recurrent models for summarization that prevent parallelization due to sequential computation, our models offer better parallelization while achieving superior performance.

- We introduce a new formulation of learning video summarization from unpaired data (Chapter 4) which is much easier to collect than the standard supervised training data. The unpaired data consists of a set of raw videos and a set of video summaries where there exists no correspondence information between these two sets. Given this unpaired data, we use adversarial learning to learn a model that yields promising results on the benchmark video summarization datasets.

- We develop an adaptive model that learns to make user-specific and personalized highlight of a video by referring to the user's highlights history (Chapter 5). We introduce a temporal-adaptive instance normalization mechanism to induce personalization signal from the user history in the model.

- We present a new architecture for sentence-guided video thumbnail generation (Chapter 6). We propose a sentence-guided temporal modulation to adapt the

video thumbnails to the user natural language sentence query. In addition, we introduce self-supervision through an auxiliary task to improve the performance of the model and also develop an unsupervised approach.

1.2 Thesis Outline

The remainder of this thesis is structured as follows. In Chapter 2, we briefly discuss the related work. In Chapter 3, we present a novel non-recurrent model for video summarization. In Chapter 4, we introduce a new formulation of learning video summarization with unpaired data. In Chapter 5, we present an adaptive video highlight detection model that produces user-specific video highlights by looking at the user's highlight history. In Chapter 6, we present a method for sentence-guided dynamic video thumbnail generation. In Chapter 7, we conclude this thesis and discuss some interesting future directions.

Chapter 2

Related Work

In this chapter, we briefly discuss the related work to this thesis. We firstly cover the related literature for the different forms of video abstraction that we study in this thesis, i.e., video summarization, video highlight detection, and dynamic video thumbnail generation. Next, we discuss the related deep learning architectures that partly inspired our models, followed by a brief description on work with weak supervision through unpaired data. Finally, we present relevant work on personalization that aims to address the issue of subjectivity in video abstraction.

2.1 Video Summarization

Video summarization aims to generate a concise overview of a video [16; 84]. Most of the prior approaches in video summarization fall in the realm of unsupervised and supervised learning.

Unsupervised methods [12; 14; 36; 39; 44; 52; 55; 57; 61; 64; 68; 79; 101; 108]

typically use hand-crafted heuristics to satisfy certain properties (e.g., diversity, representativeness, and coherency) in order to create the summary videos. Some summarization methods also provide weak supervision through additional cues such as web images/videos [9; 38; 39; 79] and video category information [63; 68] to improve the performance.

Supervised methods [23; 26; 27; 47; 58; 71; 73; 102; 103; 104; 105; 106; 110] learn video summarization from labeled data consisting of raw videos and their corresponding ground-truth summary videos. Supervised methods tend to outperform unsupervised ones, since they can learn useful cues from ground-truth summaries that are hard to capture with hand-crafted heuristics. Although supervised methods are promising, they are limited by the fact that they require expensive labeled training data in the form of videos and their corresponding summaries. This kind of training data is also referred as paired training data.

2.2 Video Highlight Detection

Different from video summarization which tries to create the overview of the video, the goal of video highlight detection is to identify moments and events in the video that a user is likely to find interesting [16].

A ranking-based learning formulation is popular in video highlight detection [16; 28; 34; 80; 96; 97]. These methods collect a set of positive and negative segments from videos. These video segments are also usually fixed in length (e.g., 5 seconds). Positive segments belong to the highlights in the ground truth, whereas segments that do not belong to highlights are denoted as negative segments. These methods sample

a fixed number of positive and negative segments from each video. A ranking-based model typically learns to score interesting/positive segments of a video higher than non-interesting/negative segments. In testing, the learned model is used to score segments in the test video. The video segments that receive high scores are included in the final highlight prediction of the video, while the low scored video segments are discarded.

Note that one major limitation of ranking-based highlight detection methods is that they typically do not process the entire video for training and learning as they operate on a fixed number of positive and negative segments on each video. Since an entire video is not processed, we believe that this approach may not efficiently understand video structure and temporal dependencies shown to be crucial in a closely related task, video summarization [103]. Another one is that they often rely on shot boundary detection algorithms (e.g., [25]) to detect shot boundaries or transitions in the videos as a preprocessing step. This results increase in both computational cost and complexity of the methods. In this thesis, we address these limitations and develop efficient models for video highlight detection.

2.3 Dynamic Video Thumbnail Generation

Traditional methods [17; 21; 29; 37; 59; 78] for video thumbnail generation operate entirely on visual features and characteristics. These methods do not consider any other information related to the video such as textual queries from the user when generating the thumbnail. Video thumbnails are obtained by extracting meaningful keyframes which has been the main focus in this line of research. For instance, Gao

et al.[21] propose a video theme-based model that selects keyframes on an underlying video theme so as to produce semantically representative thumbnails. Additionally, Song *et al.*[78] present a method that selects attractive thumbnails by examining visual aesthetic quality and relevance to the content of the video. In short, traditional methods for thumbnail generation are static and often less expressive.

Some recent methods on automatic thumbnail selection [49; 53; 99] propose to leverage user textual queries to generate video thumbnails. Most of these methods, however, do not handle complex user queries and are inspired by multi-modal semantic matching models [19; 62] that are popular for image search and tagging. The work by Yuan *et al.*[99] is the most relevant one to ours. This method combines a variant of pointer network [89], graph convolution network [41] and BiGRU [11] to address thumbnail selection based on the user query. This method produces dynamic thumbnails for videos by referring to user query sentences but is computationally complex. In this thesis, we propose a simple yet effective model for dynamic video thumbnail generation using user sentences.

Since dynamic video thumbnail generation includes natural language textual queries, it is related to the task of temporal sentence localization in videos that aims to determine the starting and ending of a continuous video segment that matches with the given natural language sentence [5; 20; 50; 98]. Unlike this task, dynamic video thumbnail generation may contain several nonconsecutive video segments [99]. Additionally, temporal sentence localization mainly focuses on matching a sentence to a video segment. In contrast, a dynamic video thumbnail should also present a quick preview of the video content along with establishing the relationship with the user

query sentence [99].

2.4 Non-recurrent Deep Learning Models

Recently, non-recurrent models, namely convolutional and self-attention models, are emerging for learning long-range dependency in sequential data (e.g., language and speech). They are computationally and statistically more efficient than recurrent models such as Long Short-Term Memory (LSTM) [30] commonly applied for problems in sequential data [7; 88].

Some recent work [43; 22; 7] in action detection, audio synthesis, and machine translation shows that convolutional networks can outperform recurrent networks. The benefit of convolutional networks is that they can process all the frames in a video simultaneously, and therefore can take the full advantage of GPU parallelization.

There is another line of research that proposes self-attention [88; 90; 100] network for long-range dependency modeling in sequences. Self-attention (also known as intra-attention) measures the response at a position in a sequence (such as language) by attending to all the positions in the given sequence [100]. Vaswani *et al.*[88] propose a self-attention model that achieves superior performance on machine translation. Wang *et al.*[90] apply self-attention to learn spatial and temporal information in video sequences.

In spite of above progress, non-recurrent models are yet to be explored in many video understanding problems including video abstraction. In this thesis, we develop non-recurrent solutions for this problem.

2.5 Learning from Unpaired Data

Recently, in image-to-image translation problem [32] (e.g., convert a grayscale image to color image) where the goal is to translate an input image to output image of different style is addressed using unpaired data [4; 111]. An unpaired training data consists of two sets: a source set (e.g., grayscale images) and a target set (e.g. color images), with no information regarding which source data matches with which target data [111]. These works take advantage of generative adversarial networks (GANs) [24] to learn the mapping between the source and target set. The key advantage of unpaired data is that it is much easier and cheaper to obtain than the paired training data used in supervised learning setting.

We find this unpaired data setting can be applied to approaches in video abstraction such as video summarization, where collecting labeled training data (videos and their summaries) is extremely expensive and time-consuming. We propose a model that leverage unpaired data for learning video summarization.

2.6 Personalized Models

Summarizing a video to present its overall synopsis or interesting moments is very challenging. To consider something in a video as relevant or interesting is subjective since different users may not share same interests [16; 77]. This may result in different summaries for the same video, implying there is no single summary possible for a video unless users interact and adapt among themselves [75]. Hence, an ideal method should aim to adapt to a user interests and preferences when summarizing a video.

Early methods propose to perform personalization using annotated meta-data [3; 6; 33; 82]. Some approaches produce personalized summaries by allowing user input (e.g., textual queries) in the learning and prediction. Sharghi *et al.* [74; 75] introduce query-focused video summarization. They consider user preference in the form of textual queries (e.g., a word, car) related to a video in their summarization framework. Vasudevan *et al.*[87] extend these works by proposing a method that is able generate video summaries based on more complicated text queries (e.g., hairstyles for men). The main limitation of these approaches is that they require the user to know the content of the video. More recently, Molino and Gygli [16] propose a model that takes a user's history as an input to make personalized predictions, making it more suitable and promising in practice as it does not expect the user to know the content of the video. However, their method is a yet another ranking model that operates on a few sample positive and negative segments of a video combined with the user's previously created highlights (i.e., user's history) for generating personalized highlights for the input video. Again, this technique is computationally expensive as it requires shot boundary detection. Furthermore, it does not process an entire video in model learning which precludes efficient capturing of long-term temporal dependencies in the video.

In this thesis, we develop a model that seamlessly learns to combine a user's preferences or interests in learning to predict personalized and user-specific video highlights. At the core of our technique is a newly designed conditional normalization scheme for videos that is partly inspired from conditional batch normalization [15] and adaptive instance normalization [31]. These methods spatially normalize the activation

from a layer to zero mean and unit variance, and then apply an affine transformation whose parameters are computed using external data [66]. These methods have been successfully applied in image understanding tasks such as visual question answering (e.g., Vries *et al.*[15]) and image-to-image translation (e.g., Liu *et al.*[51]). In contrast, we propose a conditional normalization mechanism that is suitable for videos. We demonstrate its application in adaptive video highlight detection that takes the user's history (i.e., previously created video highlights) into consideration while making a highlight prediction for an input user video.

Chapter 3

Fully Convolutional Sequence Networks

3.1 Chapter Overview and Introduction

In this chapter, we propose a novel non-recurrent model for video summarization, a common form of video abstraction. The focus in video summarization is to generate a short summary of a video such that it presents an overview of the content in the video.

We consider video summarization as a keyframe selection problem. Given an input video, our goal is to select a subset of the frames to form the summary video. Equivalently, video summarization can also be formulated as a sequence labeling problem, where each frame is assigned a binary label to indicate whether it is selected in the summary video.

Current state-of-the-art methods [58; 103] consider video summarization as a se-

quence labeling problem and solve the problem using a variant of recurrent neural networks known as the Long Short-Term Memory (LSTM) [30]. Each time step in the LSTM model corresponds to a frame in the input video. At each time step, the LSTM model outputs a binary value indicating whether this frame is selected in the summary video. The advantage of LSTM is that it can capture long-term structural dependencies among frames. But these LSTM-based models have inherent limitations. The computation in LSTM is usually left-to-right. This means we have to process one frame at a time and each frame must wait until the previous frame is processed. Although Bi-directional LSTM (Bi-LSTM) [72] exists, the computation in either direction of Bi-LSTM still suffers the same problem. Due to this sequential nature, the computation in LSTM cannot be easily parallelized to take full advantage of the GPU hardware. In our work, we propose fully convolutional models that can process all the frames simultaneously, and therefore take the full advantage of GPU parallelization. Our model is partly inspired by some recent work [43; 22; 7] in action detection, audio synthesis, and machine translation showing that convolutional models can outperform recurrent models and can take full advantage of GPU parallelization.

We propose to use fully convolutional networks for video summarization. Fully convolutional networks (FCN) [54] have been extensively used in semantic segmentation. Compared with video summarization, semantic segmentation is a more widely studied topic in computer vision. Traditionally, video summarization and semantic segmentation are considered as two completely different problems in computer vision. Our insight is that these two problems in fact share a lot of similarities. In

semantic segmentation, the input is a 2D image with 3 color channels (RGB). The output of semantic segmentation is a 2D matrix with the same spatial dimension as the input image, where each cell of the 2D matrix indicates the semantic label of the corresponding pixel in the image. In video summarization, let us assume that each frame is represented as a K-dimensional vector. This can be a vector of raw pixel values or a precomputed feature vector. Then the input to video summarization is a 1D image (over temporal dimension) with K channels. The output is a 1D matrix with the same length as the input video, where each element indicates whether the corresponding frame is selected for the summary. In other words, although semantic segmentation and video summarization are two different problems, they only differ in terms of the dimensions of the input (2D vs. 1D) and the number of channels (3 vs. K). Figure 3.1 illustrates the relationship between these two tasks. By establishing the connection between these two tasks, we can directly exploit models in semantic segmentation and adapt them for video summarization. In this chapter, we develop our video summarization method based on popular semantic segmentation models such as FCN [54]. We call our approach the *Fully Convolutional Sequence Network (FCSN)*.

FCSN is suitable for video summarization due to two important reasons. First, FCSN consist of stack of convolutions whose effective context size grows (though smaller in the beginning) as we go deeper in the network. This allows the network to model the long-range complex dependency among input frames that is necessary for video summarization. Second, FCSN is fully convolutional. Compared to LSTM, FCSN allows easier parallelization over input frames.

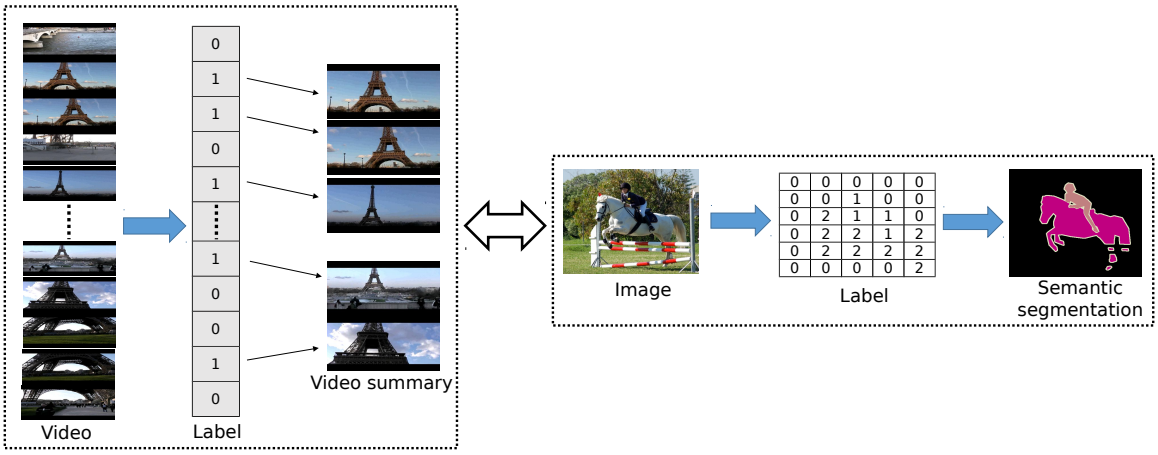

Figure 3.1: An illustration of the relationship between video summarization and semantic segmentation. (*Left*) In video summarization, our goal is to select frames from an input video to generate the summary video. This is equivalent to assigning a binary label (0 or 1) to each frame in the video to indicate whether the frame is selected for summary. This problem has a close connection with semantic segmentation (*Right*) where the goal is to label each pixel in an image with its class label.

3.2 Problem Formulation

Previous work has considered two different forms of output in video summarization: 1) binary labels; 2) frame-level importance scores. Binary label outputs are usually referred to as either keyframes [14; 23; 60; 103] or keyshots [26; 27; 68; 79; 103]. Keyframes consist of a set of non-continuous frames that are selected for the summarization, while keyshots correspond to a set of time-intervals in video where each interval consists of a continuous set of frames. Frame-level importance scores [26; 79] indicate how likely a frame should be selected for the summarization. Existing datasets have ground-truth annotations available in at least one of these two forms. Although frame-level scores provide richer information, it is practically much easier to collect annotations in terms of binary labels. It may even be possible to collect binary label annotations automatically from edited video content online. For example, if we have

access to professionally edited summary videos and their corresponding raw videos, we can automatically create annotations in the form of binary labels on frames. In this chapter, we focus on learning video summarization from only binary label-based (in particular, keyframe-based) annotations.

Let us consider a video with T frames. We assume each frame has been preprocessed (e.g., by a pretrained CNN) and is represented as a feature vector. We denote the frames in a video as $\{F_1, F_2, F_3, ..., F_T\}$ where F_i is the feature descriptor of the t-th ($t \in \{1, 2, .., T\}$) frame in the video. Our goal is to assign a binary label (0 or 1) to each of the T frames. The summary video is obtained by combining the frames that are labeled as 1 (see Fig. 3.1). We assume access to a training dataset of videos, where each frame has a ground-truth binary label indicating whether this frame should be selected in the summary video.

3.3 Architecture of FCSN

Our models are inspired by fully convolutional models used in semantic segmentation. Our models have the following properties. 1) Semantic segmentation models use 2D convolution over 2D spatial locations in an image. In contrast, our models apply 1D convolution across the temporal sequence domain. 2) Unlike LSTM models [103] for video summarization that process frames in a sequential order, our models process all frames simultaneously using the convolution operation. 3) Semantic segmentation models usually use an encoder-decoder architecture, where an image is first processed by the encoder to extract features, then the decoder is used to produce the segmentation mask using the encoded features. Similarly, our models can also be interpreted as

an encoder-decoder architecture. The encoder is used to process the frames to extract both high-level semantic features and long-term structural relationship information among frames, while the decoder is used to produce a sequence of 0/1 labels. We call our model the *fully convolutional sequence network (FCSN)*.

Our models mainly consist of temporal modules such as temporal convolution, temporal pooling, and temporal deconvolution. This is analogous to the modules commonly used in semantic segmentation models, such as 2D convolution, 2D pooling, 2D deconvolution. Due to the underlying relationship between video summarization and semantic segmentation, we can easily borrow the network architecture from existing semantic segmentation models when designing an FCSN for video summarization. In this section, we describe an FCSN based on a popular semantic segmentation network, namely FCN [54]. We refer to this FCSN as SUM-FCN. It is important to note that FCSN is certainly not limited to this particular network architecture. We can convert almost any existing semantic segmentation models into an FCSN for video summarization.

SUM-FCN: FCN [54] is a widely used model for semantic segmentation. In this section, we adapt FCN (in particular, FCN-16) for the task of video summarization. We call the model SUM-FCN. In FCN, the input is an RGB image of shape $m \times n \times 3$ where m and n are height and width of the image respectively. The output/prediction is of shape $m \times n \times C$ where the channel dimension C corresponds to the number of classes. In SUM-FCN, the input is of dimension $1 \times T \times D$ where T is the number of frames in a video and D is the dimension of the feature vector of a frame. The output of SUM-FCN is of dimension $1 \times T \times C$. Note that the dimension of the

output channel is $C = 2$ since we need scores corresponding to 2 classes (keyframe or non-keyframe) for each frame.

Figure 3.2 shows the architecture of our SUM-FCN model. We convert all the spatial convolutions in FCN to temporal convolutions. Similarly, spatial maxpooling and deconvolution layers are converted to corresponding temporal counterparts. We organize our network similar to FCN. The first five convolutional layers (*conv*1 to *conv*5) consist of multiple temporal convolution layers where each temporal convolution is followed by a batch normalization and a ReLU activation. We add a temporal maxpooling next to each convolution layer. Each of *conv*6 and *conv*7 consists of a temporal convolution, followed by ReLU and dropout. We also have *conv*8 consisting of a 1×1 convolution (to produce the desired output channel), batch normalization, and deconvolution operation along the time axis. We then take the output of *pool*4, apply a

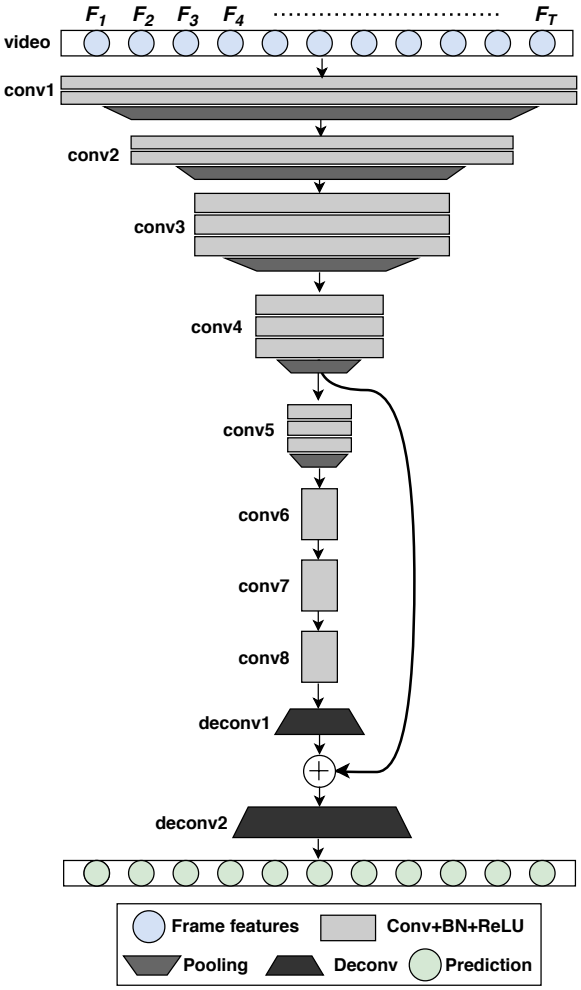

Figure 3.2: The architecture of SUM-FCN. It is based on the popular semantic segmentation architecture FCN [54]. Unlike FCN, SUM-FCN performs convolution, pooling and deconvolution operation across time.

1×1 convolution and batch normalization and then merge (element-wise addition) it with *deconv*1 feature map. This merging corresponds to the skip connection in [54]. Skip connection is widely used in semantic segmentation to combine feature maps at coarse layers with fine layers to produce richer visual features. Our intuition is that this skip connection is also useful in video summarization, since it will help in recovering temporal information required for summarization. Lastly, we apply a temporal deconvolution again and obtain the final prediction of length T.

Learning: In keyframe-based supervised setting, the classes (keyframe vs. non-keyframe) are extremely imbalanced since only a small number of frames in an input video are selected in the summary video. This means that there are very few keyframes compared with non-keyframes. A common strategy for dealing with such class imbalance is to use a weighted loss for learning. For the c-th class, we define its weight $w_c = \frac{median\_freq}{freq_c}$, where $freq_c$ is the number of frames with label c divided by the total number of frames in videos where label c is present, and $median\_freq$ is simply the median of the computed frequencies. Note that this class balancing strategy has been used for pixel labeling tasks as well [18].

Suppose we have a training video with T frames. We also have a ground-truth binary label (i.e., number of classes, C = 2) on each frame of this video. We can define the following loss $\mathcal{L}_{sum}$ for learning:

$$\mathcal{L}_{sum} = -\frac{1}{T}\sum_{t=1}^{T} w_{c_t} \log\Big(\frac{\exp(\phi_{t,c_t})}{\sum_{c=1}^{C}\exp(\phi_{t,c})}\Big), \tag{3.1}$$

where c_t is the ground-truth label of the t-th frame. $\phi_{t,c}$ and w_c indicate the score of predicting the t-th frame as the c-th class and the weight of class c, respectively.

3.4 Unsupervised SUM-FCN

In this section, we present an extension of the SUM-FCN model. We develop an unsupervised variant (called SUM-FCN$_{unsup}$) of SUM-FCN to learn video summarization from a collection of raw videos without their ground-truth summary videos.

Intuitively, the frames in the summary video should be visually diverse [103; 58]. We use this property of video summarization to design SUM-FCN$_{unsup}$. We develop SUM-FCN$_{unsup}$ by explicitly encouraging the model to generate summary videos where the selected frames are visually diverse. In order to enforce this diversity, we make the following changes to the decoder of SUM-FCN. We first select Y frames (i.e., keyframes) based on the prediction scores from the decoder. Next, we apply a 1×1 convolution to the decoded feature vectors of these keyframes to reconstruct their original feature representations. We then merge the input frame-level feature vectors of these selected Y keyframes using a skip connection. Finally, we use a 1×1 convolution to obtain the final reconstructed features of the Y keyframes such that each keyframe feature vector is of the same dimension as its corresponding input frame-level feature vector.

We use a repelling regularizer [109] $\mathcal{L}_{div}$ to enforce diversity among selected keyframes. We define $\mathcal{L}_{div}$ as the mean of the pairwise similarity between the selected Y keyframes:

$$\mathcal{L}_{div} = \frac{1}{|Y|(|Y|-1)} \sum_{t \in Y} \sum_{t' \in Y, t' \neq t} d(f_t, f_{t'}), \text{ where } (f_t, f_{t'}) = \frac{f_t^T f_{t'}}{\|f_t\|_2 \|f_{t'}\|_2} \tag{3.2}$$

and where f_t is the reconstructed feature vector of the frame t. Ideally, a diverse subset of frames will lead to a lower value of $\mathcal{L}_{div}$.

We also introduce a reconstruction loss $\mathcal{L}_{recon}$ that computes the mean squared error between the reconstructed features and the input feature vectors of the keyframes. The final learning objective of SUM-FCN$_{unsup}$ becomes $\mathcal{L}_{div} + \mathcal{L}_{recon}$. Since this objective does not require ground-truth summary videos, SUM-FCN$_{unsup}$ is an unsupervised approach.

It is worth noting that SUM-FCN will implicitly achieve diversity to some extent because it is supervised. SUM-FCN learns to mimic the ground-truth human annotations. Presumably, the ground-truth summary videos (annotated by humans) have diversity among the selected frames, since humans are unlikely to annotate two very similar frames as keyframes.

3.5 Experiments

In this section, we first introduce the datasets in Sec. 3.5.1. We then discuss the implementation details and setup in Sec. 3.5.2. Lastly, we present the main results in Sec. 3.5.3 and additional ablation analysis in Sec. 3.5.4.

3.5.1 Datasets

We evaluate our method on two benchmark datasets: SumMe [26] and TVSum [79]. The SumMe dataset is a collection of 25 videos that cover a variety of events (e.g., sports and holidays). The videos in SumMe are 1.5 to 6.5 minutes in length. The TVSum dataset contains 50 YouTube videos of 10 different categories (e.g., making sandwich, dog show, and changing vehicle tire) from the TRECVid Multimedia Event Detection (MED) task [76]. The videos in this dataset are typically 1 to 5 minutes

in length.

Since training a deep neural network with small annotated datasets is difficult, previous work [103] has proposed to use additional videos to augment the datasets. Following [103], we use 39 videos from the YouTube dataset [14] and 50 videos from the Open Video Project (OVP) dataset [14; 2] to augment the training data. In the YouTube dataset, there are videos consisting of news, sports and cartoon. In the OVP dataset, there are videos of different genres such as documentary. These datasets are diverse in nature and come with different types of annotations. We discuss in Sec. 3.5.2 on how we handle different formats of ground-truth annotations.

3.5.2 Implementation Details and Setup

Features: Following [103], we uniformly downsample the videos to 2 fps. Next, we take the output of the *pool*5 layer in the pretrained GoogLeNet [81] as the feature descriptor for each video frame. The dimension of this feature descriptor is 1024. Note that our model can be used with any feature representation. We can even use our model with video-based features (e.g., C3D [83]). We use GoogLeNet features mainly because they are used in previous work [103; 58] and will allow fair comparison in the experiments.

Ground-truth: Since different datasets provide the ground-truth annotations in various format, we follow [23; 103] to generate the single set of ground-truth keyframes (small subset of isolated frames) for each video in the datasets. These keyframe-based summaries are used for training.

To perform fair comparison with state-of-the-art methods (see Evaluation Metrics

below), we need summaries in the form of keyshots (interval-based subset of frames [26; 27; 103]) in both the final generated predictions and the ground-truth annotations for test videos. For the SumMe dataset, ground-truth annotations are available in the form of keyshots, so we use these ground-truth summaries directly for evaluation. However, keyshot annotations are missing from the TVSum dataset. TVSum provides frame-level importance scores annotated by multiple users. To convert importance scores to keyshot-based summaries, we follow the procedure in [103] which includes the following steps: 1) temporally segment a video using KTS [68] to generate disjoint intervals; 2) compute average interval score and assign it to each frame in the interval; 3) rank the frames in the video based on their scores; 4) apply the knapsack algorithm [79] to select frames so that the total length is under certain threshold, which results in the keyshot-based ground-truth summaries of that video. We use this keyshot-based annotation to get the keyframes for training by selecting the frames with the highest importance scores [103]. Note that both the keyframe-based and keyshot-based summaries are represented as $0/1$ vector of length equal to the number of frames in the video. Here, a label $0/1$ represents whether a frame is selected in the summary video. Table 3.1 illustrates the ground-truth (training and testing) annotations and their conversion for different datasets.

Training and Optimization: We use keyframe-based ground-truth annotations during training. We first concatenate the visual features of each frame. For a video with T frames, we will have an input of dimension $1 \times T \times 1024$ to the neural network. We also uniformly sample frames from each video such that we end up with $T = 320$. This sampling is similar to the fixed size cropping in semantic segmentation, where

Dataset	# annotations	Training GT	Testing GT
SumMe	15-18	frame-level scores[‡]	keyshots
TVSum	20	frame-level scores[‡]	frame-level scores †
YouTube	5	keyframes[§]	-
OVP	5	keyframes[§]	-

Table 3.1: Ground-truth (GT) annotations used during training and testing for different datasets. [‡]We convert frame-level importance scores from multiple users to single keyframes as in [79; 103]. [†]We follow [103] to convert multiple frame-level scores to keyshots. [§]Following [23; 103], we generate one set of keyframes for each video. Note that the YouTube and OVP datasets are only used to supplement the training data (as in [103; 58]), so we do not test our methods on them.

training images are usually resized to have the same spatial size. Note that our proposed model, SUM-FCN, can also effectively handle longer and variable length videos (see Sec. 3.5.4).

During training, we set the learning rate to 10^{-3}, momentum to 0.9, and batch size to 5. Other than using the pretrained GoogLeNet to extract frame features, the rest of the network is trained end-to-end using stochastic gradient descent (SGD) optimizer.

Testing: At test time, a uniformly sampled test video with $T = 320$ frames is forwarded to the trained model to obtain an output of length 320. Then this output is scaled to the original length of the video using nearest-neighbor. For simplicity, we use this strategy to handle test videos. But since our model is fully convolutional, it is not limited to this particular choice of video length. In Sec. 3.5.4, we experiment with sampling the videos to a longer length. We also experiment with directly operating

on original non-sampled (variable length) videos in Sec. 3.5.4.

We follow [103; 58] to convert predicted keyframes to keyshots so that we can perform fair comparison with other methods. We first apply KTS [68] to temporally segment a test video into disjoint intervals. Next, if an interval contains a keyframe, we mark all the frames in that interval as 1 and we mark 0 to all the frames in intervals that have no keyframes. This results in keyshot-based summary for the video. To minimize the number of generated keyshots, we rank the intervals based on the number of keyframes in intervals divided by their lengths, and finally apply knapsack algorithm [79] to ensure that the produced keyshot-based summary is of maximum 15% in length of the original test video.

Evaluation Metrics: Following [103; 58], we use a keyshot-based evaluation metric. For a given video V, suppose S_O is the generated summary and S_G is the ground-truth summary. We calculate the precision (P) and recall (R) using their temporal overlap:

$$P = \frac{|S_O \cap S_G|}{|S_O|}, R = \frac{|S_O \cap S_G|}{|S_G|} \tag{3.3}$$

Finally, we use the F-score $F = (2P \times R)/(P + R) \times 100\%$ as the evaluation metric. We follow the standard approach described in [79; 27; 103] to calculate the metric for videos that have multiple ground-truth summaries.

Experiment Settings: Similar to previous work [102; 103], we evaluate and compare our method under the following three different settings.

1. *Standard Supervised Setting*: This is the conventional supervised learning setting where training, validation and test data are drawn (such that they do not overlap)

from the same dataset. We randomly select 20% for testing and leave the remaining 80% for training and validation. Since the data is randomly split, we repeat the experiment over multiple random splits and report the average F-score performance.

2. *Augmented Setting*: For a given dataset, we randomly select 20% of data for testing and leave the remaining 80% for training and validation. In addition, we use the other three datasets to augment the training data. For example, suppose we are evaluating on the SumMe dataset, we will then have 80% of SumMe videos combined with all the videos in the TVSum, OVP, and YouTube dataset for training. Likewise, if we are evaluating on TVSum, we will have 80% of TVSum videos combined with all the videos in SumMe, OVP, and YouTube for training. Similar to the standard supervised setting, we run the experiment over multiple random splits and use the average F-score for comparison.

The idea of increasing the size of training data by augmenting it with other datasets is well-known in computer vision. This is usually referred as data augmentation. Recent methods [103; 58] show that data augmentation improves the performance. Our experimental results show similar conclusion.

3. *Transfer Setting*: This is a challenging supervised setting introduced by Zhang *et al.*[102; 103]. In this setting, the model is not trained using the videos from the given dataset. Instead, the model is trained on other available datasets and tested on the given dataset. For instance, if we are evaluating on the SumMe dataset, we will train the model using videos in the TVSum, OVP, and YouTube datasets. We then use the videos in the SumMe dataset only for evaluation. Similarly, when evaluating on TVSum, we will train on videos from SumMe, OVP, YouTube, and then test on

the videos in TVSum. This setting is particularly relevant for practical applications. If we can achieve good performance under this setting, it means that we can perform video summarization in the wild. In other words, we will be able to generate good summaries for videos from domains in which we do not have any related annotated videos during training.

3.5.3 Main Results and Comparisons

We compare the performance of our approach (SUM-FCN) with prior methods on the SumMe dataset in Table 3.2. Our method outperforms other state-of-the-art approaches by a large margin.

Dataset	Method	Supervised	Augmented	Transfer
SumMe	Gygli *et al.*[26]	39.4	–	–
	Gygli *et al.*[27]	39.7	–	–
	Zhang *et al.*[102]	40.9	41.3	38.5
	Zhang *et al.*[103] (vsLSTM)	37.6	41.6	40.7
	Zhang *et al.*[103] (dppLSTM)	38.6	42.9	41.8
	Mahasseni *et al.*[58] (supervised)	41.7	43.6	–
	Li *et al.*[45]	43.1	–	–
	SUM-FCN (ours)	**47.5**	**51.1**	**44.1**

Table 3.2: Comparison of summarization performance (F-score) between SUM-FCN and other approaches on the SumMe dataset under different settings.

Table 3.3 compares the performance of our method with previous approaches on

the TVSum dataset. Again, our method achieves state-of-the-art performance. In the *standard supervsised* setting, we outperform other approaches. In the *augmented* and *transfer* settings, our performance is comparable to other state-of-the-art methods. Note that Zhang *et al.*[103] (vsLSTM) use frame-level importance scores and Zhang *et al.*[103] (dppLSTM) use both keyframe-based annotation and frame-level importance scores. But we only use keyframe-based annotation in our method. Previous method [103] has also shown that frame-level importance scores provide richer information than binary labels. Therefore, the performance of our method on TVSum is very competitive, since it does not use frame-level importance scores during training.

Dataset	Method	Supervised	Augmented	Transfer
TVSum	Zhang *et al.*[103] (vsLSTM)	54.2	57.9	56.9†
	Zhang *et al.*[103] (dppLSTM)	54.7	59.6	**58.7**‡
	Mahasseni *et al.*[58] (supervised)	56.3	**61.2**	–
	Li *et al.*[45]	52.7	–	–
	SUM-FCN (ours)	**56.8**	59.2	58.2

Table 3.3: Performance (F-score) of SUM-FCN and other approaches on the TVSum dataset. †Zhang *et al.*[103] (vsLSTM) use frame-level importance scores. ‡Zhang *et al.*[103] (dppLSTM) use both frame-level importance scores and keyframes in their method. Different from these two methods, our method only uses keyframe-based annotations.

3.5.4 Analysis

In this section, we present additional ablation analysis on various aspects of our model.

Unsupervised SUM-FCN$_{unsup}$: Table 3.4 compares the performance of SUM-FCN$_{unsup}$ with the other unsupervised methods in the literature. SUM-FCN$_{unsup}$ achieves state-of-the-art performance on both the datasets. These results suggest that our fully convolutional sequence model can effectively learn how to summarize videos in an unsupervised way. This is very appealing since collecting labeled training data for video summarization is difficult.

Dataset	[14]	[46]	[38]	[79]	[108]	[58]	SUM-FCN$_{unsup}$
SumMe	33.7	26.6	–	26.6	–	39.1	**41.5**
TVSum	–	–	36.0	50.0	46.0	51.7	**52.7**

Table 3.4: Performance (F-score) comparison of SUM-FCN$_{unsup}$ with state-of-the-art unsupervised methods.

SUM-DeepLab: To demonstrate the generality of FCSN, we also adapt DeepLab [10] (in particular, DeepLabv2 (VGG16) model), another popular semantic segmentation model, for video summarization. We call this network SUM-DeepLab. The DeepLab model has two important features: 1) dilated convolution; 2) spatial pyramid pooling. In SUM-DeepLab, we similarly perform temporal dilated convolution and temporal pyramid pooling.

Table 3.5 compares SUM-DeepLab with SUM-FCN on the SumMe and TVSum datasets under different settings. SUM-DeepLab achieves better performance on SumMe in all settings. On TVSum, the performance of SUM-DeepLab is better than SUM-FCN in the *standard supervised* setting and is comparable in the other two settings.

We noticed that SUM-DeepLab performs slightly worse than SUM-FCN in some

settings (e.g., *transfer* setting of TVSum). One possible explanation is that the bilinear upsampling layer in DeepLab may not be the best choice. Unlike semantic segmentation, a smooth labeling (due to bilinear upsampling) is not necessarily desirable in video summarization. In other words, the bilinear upsampling may result in a sub-optimal subset of keyframes. In order to verify this, we replace the bilinear upsampling layers of SUM-DeepLab with learnable deconvolution layers (also used in SUM-FCN) and examine the performance of this modified SUM-DeepLab in the *transfer* setting. The performance of SUM-DeepLab improves as a result of this simple modification. In fact, SUM-DeepLab now achieves the state-of-the-art performance on the *transfer* setting on TVSum as well (see the last column in Table 3.5).

Dataset	Supervised	Augmented	Transfer	Transfer (deconv)
SumMe	**48.8** (47.5)	50.2 (51.1)	**45.0** (44.1)	**45.1**
TVSum	**58.4** (56.8)	59.1 (59.2)	57.4 (58.2)	**58.8**

Table 3.5: Performance (F-score) of SUM-DeepLab in different settings. We include the performance of SUM-FCN (taken from Table 3.2 and Table 3.3) in brackets. We also replace the bilinear upsampling with learnable deconvolutional layer and report the result in the transfer setting (last column).

Length of Video: We also perform experiments to analyze the performance of our models on longer-length videos. Again, we select the challenging *transfer* setting to evaluate the models when the videos are uniformly sampled to T=640 frames. Table 3.6 (first two columns) shows the results of our models for this case. Compared with $T = 320$ (shown in brackets in Table 3.6), the performance with $T = 640$ is similar.

This shows that the video length is not an issue for our proposed fully convolutional models.

Dataset	SUM-FCN T=640 (T=320)	SUM-DeepLab T=640 (T=320)	SUM-FCN variable length
SumMe	45.6 (44.1)	44.5 (45.0)	46.0
TVSum	57.4 (58.2)	57.2 (57.4)	56.7

Table 3.6: Performance (F-score) of our models on longer-length videos (i.e., T=640) and original (i.e., variable length) videos in the *transfer* data setting. In brackets, we show the performance of our model for T=320 (obtained from Tables 3.2, 3.3, and 3.5).

As mentioned earlier, the main idea behind uniformly sampling videos is to mimic the prevalent cropping strategy in semantic segmentation. Nevertheless, since our model is fully convolutional, it can also directly handle variable length videos. The last column of Table 3.6 shows the results of applying SUM-FCN (in the *transfer* setting) without sampling videos. The performance is comparable (even higher on SumMe) to the results of sampling videos to a fixed length.

Qualitative Results: In Fig. 3.3, we visualize examples of the video summarization results produced by SUM-FCN. The marked black bars on two green backgrounds are the ground truth and predicted summaries, respectively. In the first video, the content changes smoothly which results in higher performance. The second video is more challenging since it contains fast motions, multiple objects, and complex activities. The generated summaries are clustered across time and the performance is not as good as that on the first video. As future work, we believe our work can benefit from

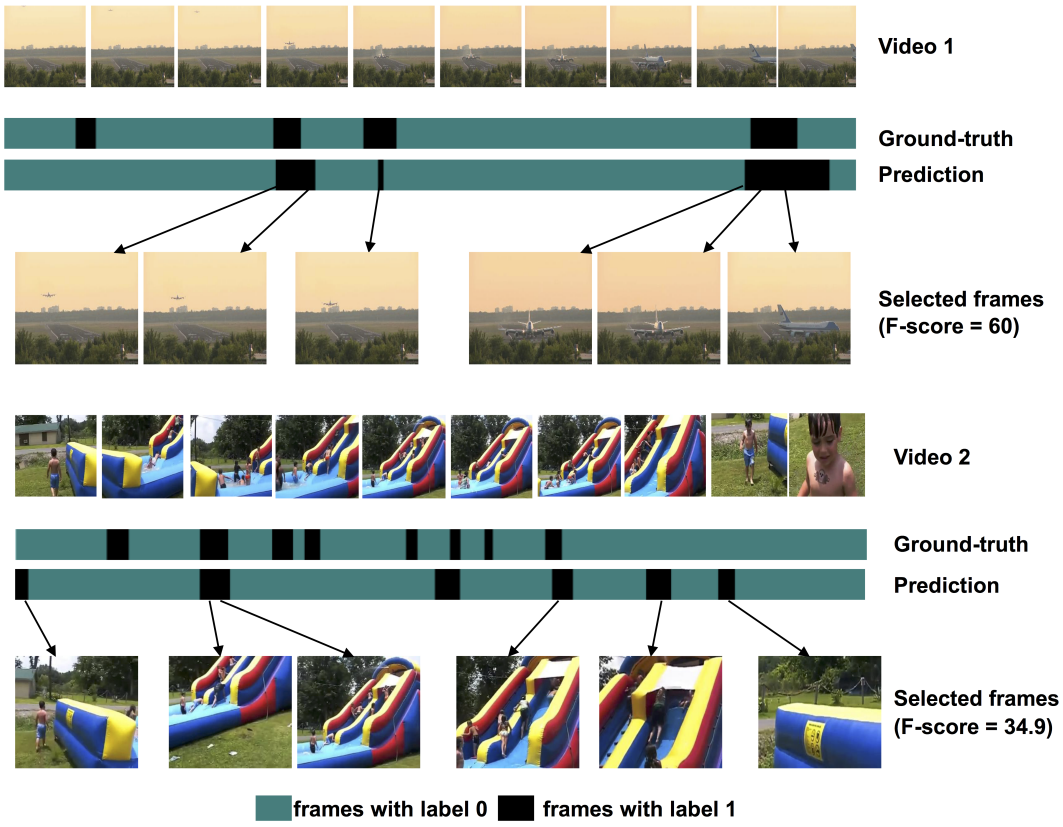

Figure 3.3: Example summaries for two videos in the SumMe [26] dataset. The black bars on the green background show the frames selected to form the summary video. For each video, we show the ground-truth (*top bar*) and the predicted labels (*bottom bar*).

incorporating higher-level semantic information, such as object detection and action recognition.

3.6 Summary

We have introduced fully convolutional sequence networks (FCSN) for video summarization. Our proposed models are inspired by fully convolutional networks in semantic segmentation. In computer vision, video summarization and semantic seg-

mentation are often studied as two separate problems. We have shown that these two seemingly unrelated problems have an underlying connection. We have adapted popular semantic segmentation networks for video summarization. Our models achieve very competitive performance in comparison with other supervised and unsupervised state-of-the-art approaches that mainly use LSTMs. We believe that fully convolutional models provide a promising alternative to LSTM-based approaches for video summarization. Finally, our proposed method is not limited to FCSN variants that we introduced. Using similar strategies, we can convert almost any semantic segmentation networks for video summarization.

Chapter 4

Learning Video Summarization from Unpaired Data

4.1 Chapter Overview and Introduction

In this chapter, we introduce a new problem formulation of learning video summarization from unpaired data, which consists of a set of raw videos and a set of video summaries that do not share any correspondence. Although we develop this formulation for video summarization, we believe this technique can be also be adapted for other approaches in video abstraction such as video highlight detection.

A major limitation of supervised video summarization is that it relies on labeled training data. Common datasets in the community are usually collected by asking human annotators to watch the input video and select the key frames or key shots. This annotation process is very expensive and time-consuming. As a result, we only have very few benchmark datasets available for video summarization in the computer

vision literature. Moreover, each dataset usually only contains a small number of annotated data (see Table 4.1).

To address the flaws of supervised learning, we propose *a new formulation of learning video summarization from unpaired data*. Our key insight is that it is much easier to collect unpaired video sets. First of all, raw videos are easily accessible as they are abundantly available on the Internet. At the same time, good summary videos are also readily available in large quantities. For example, there are lots of sports highlights, movie trailers, and other professionally edited summary videos available online. These videos can be treated as ground-truth summary videos. The challenge is that these professionally curated summary videos usually do not come with their corresponding raw input videos. In this chapter, we propose to solve video summarization by learning from such unpaired data (Fig. 4.1 (left)). We assume that our training data consist of two sets of videos: one set of raw videos (V) and another set of human created summary videos (S). However, there exists no correspondence between the videos in these two sets, i.e., the training data are *unpaired*. In other words, for a raw video in V, we may not have its corresponding ground-truth summary video in S, and vice versa.

We propose a novel approach to learn video summarization from unpaired training data. Our method learns a mapping function (called the *key frame selector*) $F : V \rightarrow S$ (Fig. 4.1 (right)) to map a raw video $v \in V$ to a summary video $F(v)$. It also trains a *summary discriminator* that tries to differentiate between a generated summary video $F(v)$ and a real summary video $s \in S$. Using an adversarial loss [24], we learn to make the distribution of generated summary videos $F(v)$ to be indistinguishable from

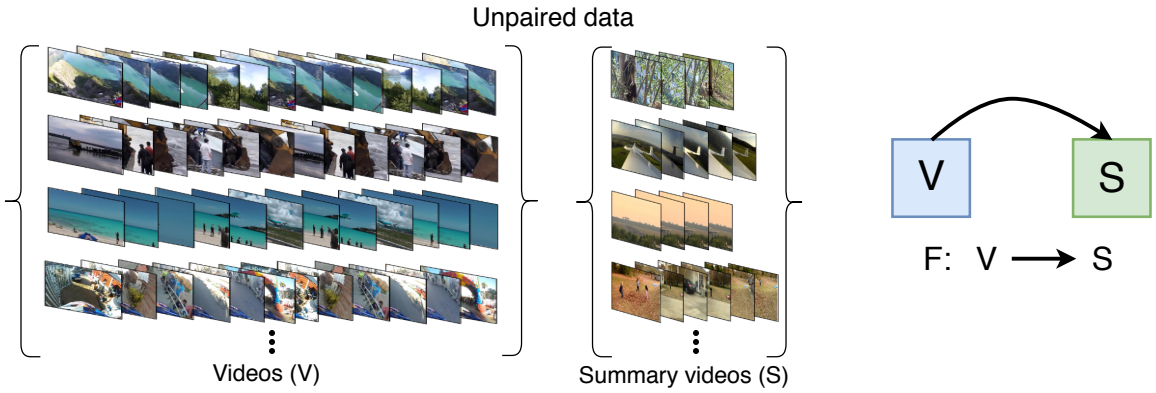

Figure 4.1: Learning video summarization from unpaired data. Given a set of raw videos $\{v_i\}_{i=1}^{M}$ $(v \in V)$ and real summary videos $\{s_j\}_{j=1}^{N}$ $(s \in S)$ such that there exists no matching/correspondence between the instances in V and S, our aim is to learn a mapping function $F : V \rightarrow S$ (right) linking two different domains V and S. The data are *unpaired* because the summary set S does not include ground truth summary videos for raw videos in V, and vice versa.

the distribution of real summary videos in S. As a result, the mapping function F will learn to generate a realistic summary video for a given input video. We also add more structure to our learning by introducing a reconstruction loss and a diversity loss on the output summary video $F(v)$. By combining these two losses with the adversarial loss, our method learns to generate meaningful and visually diverse video summaries from unpaired data.

In summary, our contributions include: (i) a new problem formulation of learning video summarization from unpaired data, which consists of a set of raw videos and a set of video summaries that do not share any correspondence; (ii) a deep learning model for video summarization that learns from unpaired data via an adversarial process; (iii) an extensive empirical study on benchmark datasets to demonstrate the effectiveness of the proposed approach; and (iv) an extension of our method that introduces partial supervision to improve the summarization performance.

4.2 Our Approach

4.2.1 Formulation

We are given an unpaired dataset consisting of a set of raw videos $\{v_i\}_{i=1}^{M}$ and a set of real summary videos $\{s_j\}_{j=1}^{N}$, where $v_i \in V$ and $s_j \in S$. We define the data distribution for v and s as $v \sim p_{data}(v)$ and $s \sim p_{data}(s)$, respectively. Our model consists of two sub-networks called the *key frame selector network* (S_K) and the *summary discriminator network* (S_D). The key frame selector network is a mapping function $S_K : V \rightarrow S$ between the two domains V and S (see Fig. 4.1). Given an input video $v \in V$, the key frame selector network (S_K) aims to select a small subset of k key frames of this video to form a summary video $S_K(v)$. The goal of the summary discriminator network (S_D) is to differentiate between a real summary video $s \in S$ and the summary video $S_K(v)$ produced by the key frame selector network S_K. Our objective function includes an adversarial loss, a reconstruction loss and a diversity loss. We learn the two networks S_K and S_D in an adversarial fashion. In the end, S_K learns to output an optimal summary video for a given input video. In practice, we precompute the image feature of each frame in a video. With a little abuse of terminology, we use the term "video" to also denote the sequence of frame-level feature vectors when there is no ambiguity based on the context.

4.2.2 Network Architecture

The *key frame selector network* (S_K) in our model takes a video with T frames as the input and produces the corresponding summary video with k key frames. We

use the fully convolutional sequence network (FCSN) [71], an encoder-decoder fully convolutional network, to select key frames from the input video. FCSN encodes the temporal information among the video frames by performing convolution and pooling operations in the temporal dimension. This enables FCSN to extract representations that capture the inter-frame structures. The decoder of FCSN consists of several temporal deconvolution operations which produces a vector of prediction scores with the same length as the input video. Each score indicates the likelihood of the corresponding frame being a key frame or non-key frame. Based on these scores, we select k key frames to form the predicted summary video. In order to define the reconstruction loss used in the learning (see Sec. 4.2.3), we apply convolution operations on the decoded feature vectors of these k key frames to reconstruct the corresponding feature vectors in the input video. We also introduce a skip connection that retrieves the frame-level feature representation of the selected k key frames, which we merge with the reconstructed features of the k key frames. Fig. 4.2 (a) shows the architecture of S_K.

The *summary discriminator network* (S_D) in our model takes two kinds of input: (1) the summary videos produced by S_K for the raw videos in V; and (2) the real summary videos in S. The goal of S_D is to distinguish between the summaries produced by S_K and the real summaries. We use the encoder of FCSN [71] to encode the temporal information within the input summary video. Next, we perform a temporal average pooling operation (Ω_t) on the encoded feature vectors to obtain a video-level feature representation. Finally, we append a fully connected layer ($\mathcal{FC}$), followed by a sigmoid operation (σ) to obtain a score ($\mathcal{R}_s$) indicating whether the input sum-

mary video is a real summary or a summary produced by S_K. Let s be an input summary video to S_D, we can express the operations in S_D by Eq. 4.1. The network architecture of S_D is shown in Fig. 4.2 (b).

$$\mathcal{R}_s = S_D(s) = \sigma\left(\mathcal{FC}\left(\Omega_t\left(FCSN_{enc}(s)\right)\right)\right) \tag{4.1}$$

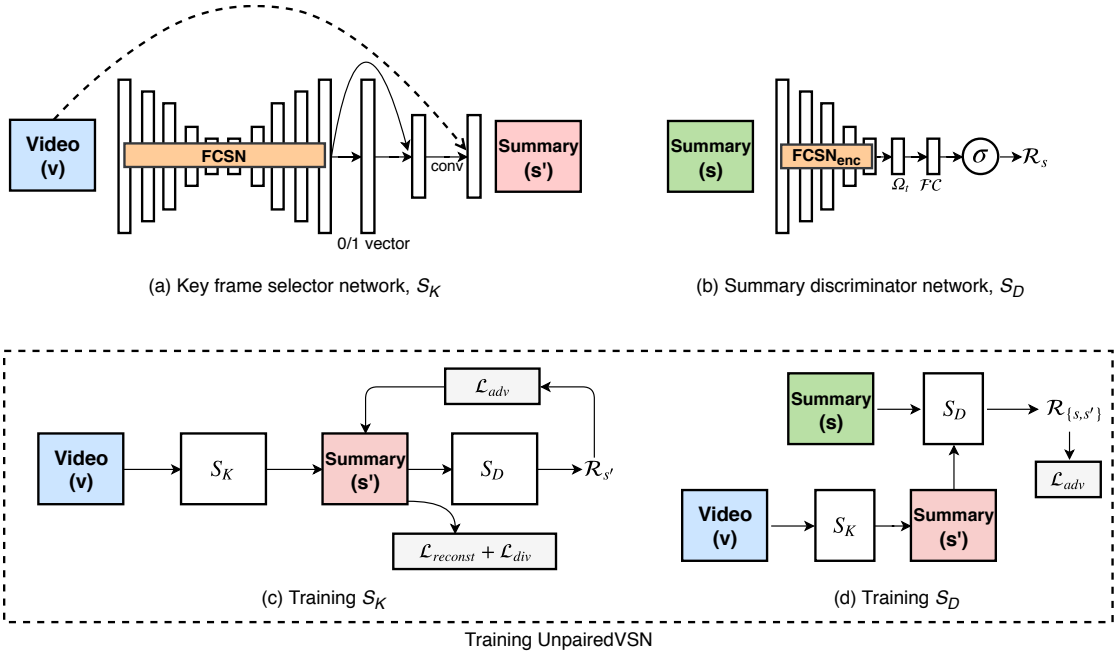

Figure 4.2: Overview of our proposed model. (a) Network architecture of the key frame selector network S_K. It takes a video v and produces its summary video s' (i.e., $S_K(v)$) by selecting k key frames from v. The backbone of S_K is FCSN [71]. We also introduce a skip connection from the input to retrieve the frame-level features of k key frames selected by S_K. (b) Network architecture of the summary discriminator network S_D. It differentiates between an output summary video s' and a real summary video s. S_D consists of the encoder of FCSN ($FCSN_{enc}$), followed by a temporal average pooling (Ω_t) and sigmoid (σ) operations. In (c) and (d), we show the training scheme of S_K and S_D, respectively. S_K tries to produce video summaries that are indistinguishable from real video summaries created by humans, whereas S_D tries to differentiate real summary videos from the summaries produced by S_K. As mentioned in Sec. 4.2.1, there is no correspondence information available to match raw videos and summary videos in the training data.

4.2.3 Learning

Our learning objective includes an adversarial loss [24], a reconstruction loss, and a diversity loss.

Adversarial Loss: This loss aims to match the distribution of summary videos produced by the key frame selector network S_K with the data distribution of real summary videos. We use the adversarial loss commonly used in generative adversarial networks [24]:

$$\begin{aligned}\mathcal{L}_{adv}(S_D, S_K) &= \mathbb{E}_{s\sim p_{data}(s)}[\log S_D(s)] \\ &\quad +\mathbb{E}_{v\sim p_{data}(v)}[\log(1 - S_D(S_K(v)))],\end{aligned} \tag{4.2}$$

where S_K aims to produce summary videos $S_K(v)$ that are close to real summary videos in domain S, and S_D tries to differentiate between output summary videos $\{S_K(v) : v \in V\}$ and real summary videos $\{s : s \in S\}$. A minimax game occurs between S_K and S_D, where S_K pushes to minimize the objective and S_D aims to maximize it. This is equivalent to the following:

$$\min_{S_K} \max_{S_D} \mathcal{L}_{adv}(S_D, S_K). \tag{4.3}$$

Reconstruction Loss: We introduce a reconstruction loss to minimize the difference between the reconstructed feature representations of the k key frames in the predicted summary video $S_K(v)$ and the input frame-level representation of those k key frames in the input video v. Let Λ_K be a set of k indices indicating which k frames in the input video are selected in the summary. In other words, if $f \in \Lambda_k$, the f-th frame in the input video is a key frame. We can define this reconstruction loss as:

$$\mathcal{L}_{reconst}(S_K(v), v) = \frac{1}{k}\sum_{t=1}^{k} \|S_K(v)^t - v^{f_t}\|_2^2, \tag{4.4}$$

where $S_K(v)^t$ and v^{f_t} are the features of the t-th frame in the output summary video $S_K(v)$ and the f_t-th frame (i.e., $f_t \in \Lambda_k$) of the input video v, respectively. The intuition behind this loss is to make the reconstructed feature vectors of the key frames in the summary video $S_K(v)$ similar to the feature vectors of those frames in the input video v.

Diversity Loss: It is desirable in video summarization that the frames in the summary video have high visual diversity [58; 103; 110]. To enforce this constraint, we apply a repelling regularizer [109] that encourages the diversity in the output summary video $S_K(v)$ for the given input video v. This diversity loss is defined as:

$$\mathcal{L}_{div}(S_K(v)) = \frac{1}{k(k-1)} \sum_{t=1}^{k} \sum_{t'=1, t' \neq t}^{k} \frac{(S_K(v)^t)^T \cdot S_K(v)^{t'}}{\|S_K(v)^t\|_2 \|S_K(v)^{t'}\|_2}, \tag{4.5}$$

where $S_K(v)^t$ is the frame-level reconstructed feature representation of frame t in the summary video $S_K(v)$. We aim to minimize $\mathcal{L}_{div}(S_K(v))$, so that the selected k key frames are visually diverse.

Final Loss: Our final loss function is:

$$\begin{aligned} \mathcal{L}(S_K, S_D) &= \mathcal{L}_{adv}(S_D, S_K) + \mathcal{L}_{reconst}(S_K(v), v) \\ &\quad + \beta\, \mathcal{L}_{div}(S_K(v)), \end{aligned} \tag{4.6}$$

where β is a hyperparameter that controls the relative importance of the visual diversity. The goal of the leaning is to find the optimal parameters $\Theta^*_{S_K}$ and $\Theta^*_{S_D}$ in S_K and S_D, respectively. We can express this as the following:

$$\Theta^*_{S_K}, \Theta^*_{S_D} = \arg\min_{\Theta_{S_K}, \Theta_{S_D}} \mathcal{L}(S_K, S_D). \tag{4.7}$$

For brevity, we use `UnpairedVSN` to denote our unpaired video summarization

network that is learned by Eq. 4.7. In Fig. 4.2(c) and Fig. 4.2(d), we show the training scheme of S_K and S_D in our model `UnpairedVSN`.

4.2.4 Learning with Partial Supervision

In some cases, we may have a small amount of paired videos during training. We use V_p ($V_p \subset V$) to denote this subset of videos for which we have the ground truth summary videos. Our model can be easily extended to take advantage of this partial supervision. In this case, we apply an additional objective $\mathcal{L}_{psup}$ on the output of FCSN in the key frame selector network S_K. Suppose a training input video $v \in V_p$ has T frames, $\delta_{t,l}$ is the score of the t-th frame to be the l-th class (key frame or non-key frame) and l_t is the ground-truth binary key frame indicator. We define $\mathcal{L}_{psup}(v)$ as:

$$\mathcal{L}_{psup}(v) = -\frac{1}{T}\sum_{t=1}^{T} \log\Big(\frac{\exp(\delta_{t,l_t})}{\sum_{l=1}^{2}\exp(\delta_{t,l})}\Big). \tag{4.8}$$

Our learning objective in this case is defined as:

$$\begin{aligned}\mathcal{L}(S_K, S_D) &= \mathcal{L}_{adv} + \mathcal{L}_{reconst} + \beta\,\mathcal{L}_{div} \\ &\quad +\gamma \cdot 1(v) \cdot \mathcal{L}_{psup},\end{aligned} \tag{4.9}$$

where $1(\cdot)$ is an indicator function that returns 1 if $v \in V_p$, and 0 otherwise. This means that $\mathcal{L}_{psup}$ is considered if the video v is an instance in V_p for which we have the ground-truth summary video. The hyperparameters β and γ control the relative importance of the diversity and supervision losses, respectively. We denote this variant of our model as $\texttt{UnpairedVSN}_{psup}$.

4.3 Experiments

4.3.1 Setup

Data and Setting: We conduct evaluation on two standard video summarization datasets: SumMe [26] and TVSum [79]. These datasets have 25 and 50 videos, respectively. Since these datasets are very small, we use another two datasets, namely the YouTube [14] (39 videos) and the OVP dataset [2] (50 videos), to help the learning. Table 4.1 shows the main characteristics of the datasets. We can observe that these datasets are diverse, especially in terms of ground-truth annotations. We follow prior work [23; 103] to convert multiple ground truths with different format to generate a single keyframe-based annotation (a binary key frame indicator vector [103]) for each training video.

Dataset	# videos	Video types	Ground-truth annotation type
SumMe [26]	25	User	Interval-based shots and frame-level score
TVSum [79]	50	YouTube	Frame-level importance score
YouTube [14]†	39	Web videos	Collection of key frames
OVP [2]	50	Various genre	Collection of key frames

Table 4.1: Key characteristics of different datasets used in our experiments. †The YouTube dataset has 50 videos, but we exclude (following [23; 103]) the 11 cartoon videos and keep the rest.

From Table 4.1, we can see that we have in total 164 videos available for experiments. When evaluating on the SumMe dataset, we randomly select 20% of SumMe videos for testing. We use the remaining 80% of SumMe videos and all the videos in

other datasets (i.e., TVSum, YouTube, and OVP) for training. We create the *unpaired* data from the training subset by first randomly selecting 50% of the raw videos (ignoring their ground-truth summaries) and then selecting the ground-truth summaries (while ignoring the corresponding raw videos) of the remaining 50% videos. In the end, we obtain a set of raw videos and a set of real summary videos, where there is no correspondence between the raw videos and the summary videos. We follow the same strategy to create the training (unpaired) and testing set when evaluating on the TVSum dataset.

Features: Firstly, we uniformly downsample every video to 2 fps. Then we use *pool*5 layer of the pretrained GoogLeNet [81] to extract 1024-dimensional feature representation of each frame in the video. Note that our feature extraction follows prior work [58; 71; 103; 110]. This allows us to perform a fair comparison with these works.

Training Details: We train our final model (`UnpairedVSN`) from scratch with a batch size of 1. We use the Adam optimizer [40] with a learning rate of 0.00001 for the key frame selector network (S_K). We use the SGD [8] optimizer with a learning rate of 0.0002 for the summary discriminator network (S_D). We set $\beta = 1$ for SumMe and $\beta = 0.001$ for TVSum in Eq. 4.6. Additionally, we set β and γ to 0.001 for SumMe and TVSum in Eq. 4.9.

Evaluation Metrics: We evaluate our method using the keyshot-based metrics as in previous work [58; 103]. Our method predicts summaries in the form of key frames. We convert these key frames to key shots (i.e., an interval-based subset of video frames [26; 27; 103]) following the approach in [103]. The idea is to first temporally segment

the videos using KTS algorithm [68]. If a segment contains a key frame, we mark all the frames in that segment as 1, and 0 otherwise. This process may result in many key shots. In order to reduce the number of key shots, we rank the segments according to the ratio between the number of key frames and the length of segment. We then apply knapsack algorithm to generate keyshot-based summaries that are at most 15% of the length of the test video [26; 27; 79; 103]. The SumMe dataset has keyshot-based ground-truth annotation, so we directly use it for evaluation. The TVSum dataset provides frame-level importance scores which we also convert to key shots as done by [58; 103] for evaluation.

Given a test video v, let X and Y be the predicted key shot summary and the ground-truth summary, respectively. We compute the precision (P), recall (R) and F-score (F) to measure the quality of the summary as follows:

$$P = \frac{\text{overlap in } X \text{ and } Y}{\text{duration of } X}, R = \frac{\text{overlap in } X \text{ and } Y}{\text{duration of } Y} \tag{4.10}$$

$$F = \frac{2 \times P \times R}{P + R} \tag{4.11}$$

We follow the evaluation protocol of the datasets (SumMe [26; 27] and TVSum [79]) to compute the F-score between the multiple user created summaries and the predicted summary for each video in the datasets. Following prior work [58], we run our experiments five times for each method and report the average performance over the five runs.

4.3.2 Baselines

Since our work is the first attempt to learn video summarization using unpaired data, there is no prior work that we can directly compare with. Nevertheless, we

define our own baselines as follows:

Unsupervised SUM-FCN: If we remove the summary discriminator network from our model, we can learn video summarization in an unsupervised way. In this case, our learning objective is simply $\mathcal{L}_{reconst} + \mathcal{L}_{div}$. This is equivalent to the unsupervised SUM-FCN in [71]. We call this baseline model $\texttt{SUM-FCN}_{unsup}$. Note that $\texttt{SUM-FCN}_{unsup}$ is a strong baseline (as shown in [71]) since it already outperforms many existing unsupervised methods ([14; 38; 46; 58; 79; 108]) in the literature.

Model with Adversarial Objective: We define another baseline model where we have the summary discriminator network S_D and the key frame selector network S_K, but the objective to be minimized is $\mathcal{L}_{reconst} + \mathcal{L}_{adv}$ (i.e., we ignore $\mathcal{L}_{div}$). We refer to this baseline model as $\texttt{UnpairedVSN}_{adv}$.

4.3.3 Main Results

In Table 4.2, we provide the results (in terms of F-score, precision and recall) of our final model `UnpairedVSN` and the baseline models on the SumMe dataset. Our method outperforms the baseline methods on all evaluation metrics. It is also worth noting that when our summary generator and discriminator networks are trained using unpaired data with the adversarial loss (i.e., $\texttt{UnpairedVSN}_{adv}$), we observe a significant boost in performance (1.7%, 1.1% and 2.9% in terms of F-score, precision and recall, respectively) over the unsupervised baseline $\texttt{SUM-FCN}_{unsup}$. Adding an additional regularizer $\mathcal{L}_{div}$ (i.e., `UnpairedVSN`) further improves the summarization performance.

Table 4.3 shows the performance of different methods on the TVSum dataset.

Again, our final method outperforms the baseline methods. Moreover, the trend in performance boost is similar to what we observe on the SumMe dataset.

	SUM-FCN$_{unsup}$ [71]	UnpairedVSN$_{adv}$	UnpairedVSN
F-score	44.8	46.5	**47.5**
Precision	43.9	45.0	**46.3**
Recall	46.2	49.1	**49.4**

Table 4.2: Performance (%) of different methods on the SumMe dataset [26]. We report summarization results in terms of three standard metrics including F-score, Precision and Recall.

	SUM-FCN$_{unsup}$ [71]	UnpairedVSN$_{adv}$	UnpairedVSN
F-score	53.6	55.3	**55.6**
Precision	59.1	61.0	**61.1**
Recall	49.1	50.6	**50.9**

Table 4.3: Performance (%) of different methods on TVSum [79].

Results in Table 4.2 and Table 4.3 demonstrate that learning from unpaired data is advantageous as it can significantly improve video summarization models over purely unsupervised approaches.

4.3.4 Comparison with Supervised Methods

We also compare the performance of our method with state-of-the-art supervised methods for video summarization. Recent supervised methods [58; 71; 102; 103;

104; 107; 110] also use additional datasets (i.e., YouTube and OVP) to increase the number of paired training examples while training on the SumMe or the TVSum dataset. For example, when experimenting on SumMe, they use 20% for testing and use the remaining 80% videos of SumMe along with the videos in TVSum, OVP and YouTube for training. However, the main difference is that we further divide the combined training dataset to create unpaired examples (see Sec. 4.3.1). In other words, given a pair of videos (a raw video and its summary video), we either keep the raw video or the summary video in our training set. In contrast, both videos are part of the training set in supervised methods. As a result, supervised methods use twice as many videos during training. In addition, supervised methods have access to the correspondence between the raw video and the ground-truth summary video. Therefore, it is important to note that the supervised methods utilize far more supervision than our proposed method. We show the comparison in Table 4.4.

Surprisingly, on the SumMe dataset, our final method outperforms most of the supervised methods (except [71]) by a big margin (nearly 3%). On the TVSum dataset, we achieve slightly lower performance. Our intuition is that if we have more unpaired data for training, we can reduce the performance gap on TVSum. To sum up, this comparison study demonstrates that our unpaired learning formulation has potential to compete with supervised approaches.

4.3.5 Effect of Partial Supervision

We also examine the performance of our model when direct supervision (i.e., correspondence between videos in V and S) is available for a small number of videos

Method	SumMe	TVSum
Zhang *et al.*[102]	41.3	–
Zhang *et al.*[103] (vsLSTM)	41.6	57.9
Zhang *et al.*[103] (dppLSTM)	42.9	59.6
Mahasseni *et al.*[58] (supervised)	43.6	61.2
Zhao *et al.*[107]‡	43.6	61.5
Zhou *et al.*[110] (supervised)	43.9	59.8
Zhang *et al.*[104]	44.1	63.9
Rochan *et al.*[71]	51.1	59.2
`UnpairedVSN` (Ours)	47.5	55.6

Table 4.4: Quantitative comparison (in terms of F-score %) between our methods and state-of-the-art supervised methods on SumMe [26] and TVSum [79]. ‡Results are taken from [104].

in the training set. Our aim is to study the effect of adding partial supervision to the framework.

In this case, for the first 10% of original/raw videos that are fed to the key frame selector network, we use their ground-truth key frame annotations as an additional learning signal (see Eq. 4.9). Intuitively, we should be able to obtain better performance than learning only with unpaired data, since we have some extra supervision during training.

Table 4.5 shows the performance of our model trained with this additional partial supervision. We observe a trend of improvement (across all evaluation metrics) on both the datasets. This shows that our proposed model can be further improved if

we have access to some paired data in addition to unpaired data during the training.

	SumMe	TVSum
F-score	**48.0** (47.5)	**56.1** (55.6)
Precision	**46.7** (46.3)	**61.7** (61.1)
Recall	**49.9** (49.4)	**51.4** (50.9)

Table 4.5: Performance (%) of $\texttt{UnpairedVSN}_{psup}$ on the SumMe [26] and TVSum [79] datasets. In the bracket, we include the performance of our final model `UnpairedVSN` reported in Table 4.2 and Table 4.3 to help with the comparison.

4.3.6 Transfer Data Setting

In our standard data setting (see Sec. 4.3.1), it is possible that some of the unpaired examples consist of raw videos or video summaries from the dataset under consideration. In order to avoid this, we conduct additional experiments under a more challenging data setting where the unpaired examples originate totally from different datasets. For instance, if we evaluate on SumMe, we use the videos and user summaries of TVSum, OVP and YouTube to create unpaired training data, and then use the entire SumMe for testing. We follow the similar process while evaluating on TVSum. This kind of data setting is referred as transfer data setting [102; 103], though it has been defined in the context of fully supervised learning. We believe that this data setting is closer to real scenarios, where we may need to summarize videos from domains that are different from those used in training.

Table 4.6 and Table 4.7 show the performance of different approaches on SumMe and TVSum, respectively. Although we notice slight degradation in performance

compared with the standard data setting, the trend in results is consistent with our findings in Sec. 4.3.3.

	SUM-FCN$_{unsup}$ [71]	UnpairedVSN$_{adv}$	UnpairedVSN
F-score	39.5	41.4	**41.6**
Precision	38.3	40.4	**40.5**
Recall	41.2	43.6	**43.7**

Table 4.6: Performance (%) of different methods on SumMe [26] under transfer data setting.

	SUM-FCN$_{unsup}$ [71]	UnpairedVSN$_{adv}$	UnpairedVSN
F-score	52.9	55.0	**55.7**
Precision	58.2	60.6	**61.2**
Recall	48.5	50.4	**51.1**

Table 4.7: Performance (%) of different methods on TVSum [79] under transfer data setting.

4.3.7 Qualitative Analysis

Figure 4.3 presents example summaries generated by our method UnpairedVSN. We observe that the output summaries from our approach have a higher overlap with the human generated summaries. This implies that our method is able to preserve information essential for generating optimal and meaningful summaries.

We compare the results of different approaches in Fig. 4.4. The first video in Fig. 4.4(a) is related to cooking. SUM-FCN$_{unsup}$ extracts the shots from the middle of

the video and misses the important video shots towards the end. In contrast, we observe that `UnpairedVSN` preserves the temporal story of the video by extracting video shots from different sections while focusing on key scenes. This has resulted in better agreement with the human created summaries. The second video in Fig. 4.4(b) is about scuba diving. Unlike the first video, there is not a huge performance gap between `SUM-FCN`$_{unsup}$ and `UnpairedVSN`. However, it still noticeable that `SUM-FCN`$_{unsup}$ captures less diverse scenes compared with `UnpairedVSN`.

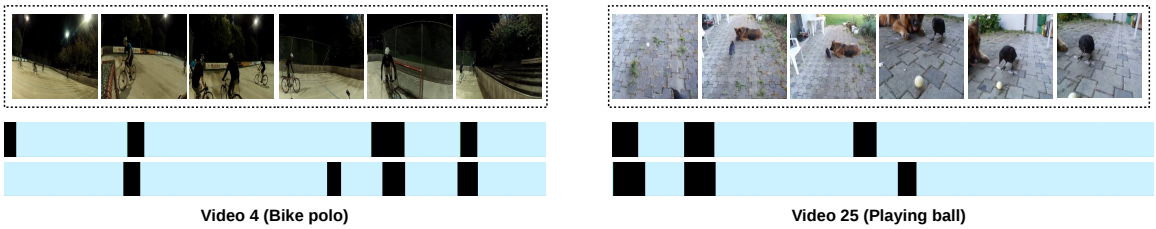

Figure 4.3: Two example results from the SumMe dataset [26]. The two bars at the bottom show the summaries produced by `UnpairedVSN` and humans, respectively. The black bars denote the selected sequences of frames, and the blue bar in background indicate the video length.

4.4 Summary

We have presented a new formulation for video summarization where the goal is to learn video summarization using unpaired training examples. We have introduced a deep learning framework that operates on unpaired data and achieves much better performance than the baselines. Our proposed method obtains results that are even comparable to the state-of-the-art supervised methods. If a small number of paired videos are available during training, our proposed framework can be easily extended to take advantage of this extra supervision to further boost the performance. Since

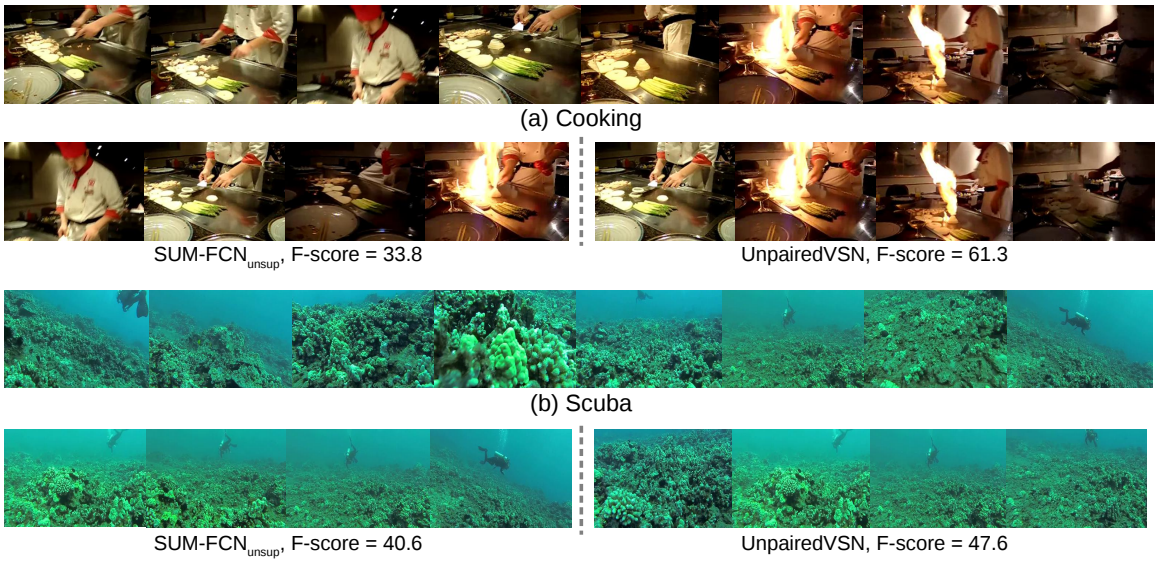

Figure 4.4: Example videos from SumMe [26] and predicted summaries by `SUM-FCN`$_{unsup}$ [71] and `UnpairedVSN`. Frames in the first row are sampled from the video, whereas frames in the second row are sampled from the summaries generated by different approaches.

unpaired training data are much easier to collect, our work offers a promising direction for future research in video summarization. As future work, we plan to experiment with large-scale unpaired videos collected in the wild. Moreover, we also aim to study the application of proposed approach to other approaches in video abstraction such as highlight detection.

Chapter 5

Learning to Adapt Video Highlight Detection using User History

5.1 Chapter Overview and Introduction

In this chapter, we propose an adaptive video highlight detection model that leverages the user's previously created video highlights to produce a user-specific highlight of an input user video.

There is a proliferation in the amount of video data captured and shared everyday. It has given rise to multifaceted challenges, including editing, indexing and browsing of this massive amount of video data. This has drawn attention of the research community to build automated video highlight detection tools. The goal of highlight detection is to reduce an unedited video to its interesting visual moments and events. A robust highlight detection solution can enhance video browsing experience by providing quick video preview, facilitating video sharing on social media and assisting

video recommendation systems.

Even though we have made significant progress in highlight detection, the existing methods are missing the ability to adapt its predictions to users. The main thrust of research in highlight detection has been on building generic models. However, different users have different preferences in term of detected highlights [16; 77]. Generic highlight detection models ignore the fact that the definition of a video highlight is inherently subjective and depends on each individual user's preference. This can greatly limit the adoption of these models in real-world applications. In Fig. 5.1, we illustrate the subjective nature of highlights. The input video contains events such as cycling, cooking, and eating. A generic highlight detection model mainly predicts the cycling event as the highlight. But if we examine the user's history (previously created highlights by the user), we can infer that this user is interested in cooking scenes. Therefore, a highlight detection model should predict the cooking event as the highlight instead. Motivated by this observation, we propose *an adaptive highlight detection model* that explicitly takes user's history into consideration while generating highlights.

Although a user's visual highlight history can provide a stronger and more reliable cue of their interests than non-visual meta-data [16], there is very limited research on adapting highlight detection using this visual information. To the best of our knowledge, the recent work by Molino and Gygli [16] is the only prior work on this topic. Their method considers a user's previously created highlights (available as GIFs[1]) from multiple videos when generating new highlights for that user. They propose a ranking model that predicts a higher score for interesting video segments

[1]GIF is an image format with multiple frames played in a loop without sound [28].

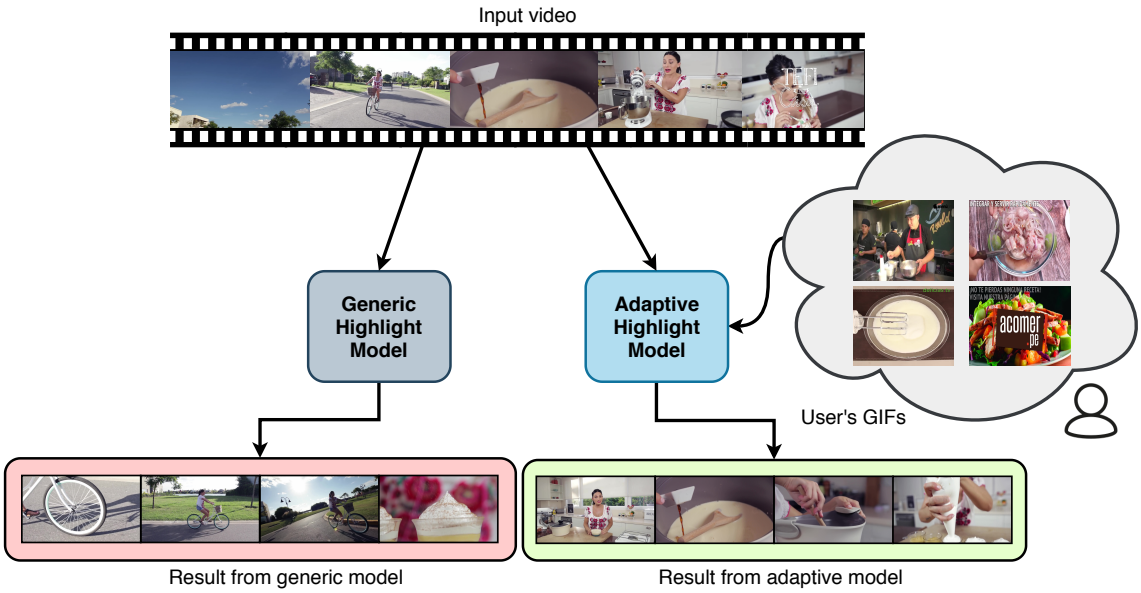

Figure 5.1: The definition of highlight of a video is inherently subjective and depends on each user's preference. In contrast to a generic highlight detection model, an adaptive highlight detection model (like ours) incorporates a user's previously created highlights (e.g., GIFs from multiple videos) when predicting highlights of an input video. This allows the model to make more accurate and user-specific highlight predictions.

as opposed to non-interesting ones while conditioning on the user's past highlights (i.e., user's history). However, their method has some limitations. Firstly, it operates at the segment-level and samples a fixed number of positive and negative segments from a video for learning. This means that the method does not process an entire video which is essential to capture temporal dependencies shown to be vital in many video understanding tasks [43; 58; 71; 103]. Moreover, it is sensitive to the number of positive and negative samples used in the learning. Secondly, it requires precomputing shot boundaries using a shot detection algorithm [25] for sampling a set of positive/negative video segments. This makes their pipeline computationally complex and expensive. Lastly, their model directly combines user's history features

with the features of sampled video segments to predict user-specific highlights. We demonstrate in our experiments that this is not as effective as our proposed model.

In this chapter, we introduce a novel user-adaptive highlight detection framework that is simple and powerful. Given an input user video for highlight detection and the user's history (highlight GIFs from multiple videos that the user has previously created), our model seamlessly combines the user's history information with the input video to modulate the highlight detection so as to make user-specific and more precise highlight prediction. Our framework consists of two sub-networks: a fully temporal convolutional *highlight detection network* H which produces the highlight for an input video, and a *history encoder network* M that encodes the user's history information. We propose *temporal-adaptive instance normalization* (T-AIN), a conditional temporal instance normalization layer for videos. We introduce T-AIN layers to H where the interaction between the two sub-networks H and M takes place. T-AIN layers have affine transformation parameters that are predicted from M based on the user's history. In other words, M acts a guiding network to H. Through the adjustable affine parameters in T-AIN layers, H can adapt highlight predictions to different users based on their preferences as expressed in their histories. Note that our method does not require expensive shot detection. Moreover, it can utilize an entire video for learning instead of a few sampled video segments.

To summarize, the main contributions of this chapter are the following. (1) We study user-adaptive highlight detection using user history in the form of previously created highlights by the user. This problem has many commercial applications, but is not well studied in the literature. (2) Different from ranking models [28;

16; 95; 96] commonly used in highlight detection, we are first to employ a fully temporal convolutional model in highlight detection. Our model does not require expensive shot detection algorithm and can process an entire video at once. This makes our proposed model simpler operationally. (3) We propose temporal-adaptive instance normalization layer (T-AIN) for videos. T-AIN layers have adjustable affine parameters which allow the highlight detection network to adapt to different users. (4) We experiment on a large-scale dataset and demonstrate the effectiveness of our approach. (5) We further explore the application of our model as a pre-trained model for video summarization, a task closely related to highlight detection.

5.2 Our Approach

Given a video $\mathbf{v}$, we represent each frame in the video as a D-dimensional feature vector. The video can be represented as a tensor of dimension $1 \times T \times D$, where T and D are the number of frames and dimension of frame feature vector of each frame in the video, respectively. T varies for different videos. For a user U, let $\mathcal{H} = \{h_1, ..., h_n\}$ denotes the *user's history* which is a collection of visual highlights that the user has created in the past.

Given $\mathbf{v}$ and $\mathcal{H}$, our goal is to learn a mapping function G that predicts two scores for each frame in $\mathbf{v}$ indicating it being non-highlight and highlight. Thus, the final output S is of dimension $1 \times T \times 2$ for the input video $\mathbf{v}$:

$$S = G(\mathbf{v}, \mathcal{H}) = G(\mathbf{v}, \{h_1, ..., h_n\}). \tag{5.1}$$

We refer to G as the adaptive highlight detector.

5.2.1 Background: Temporal Convolution Networks

In recent years, temporal convolution networks (e.g., FCSN [71]) have shown to achieve impressive performance on video understanding tasks. These networks mainly perform 1D operations (e.g., 1D convolution, 1D pooling) over the temporal dimension (e.g., over frames in a video). This is analogous to the 2D operations (e.g., 2D convolution, 2D pooling) commonly used in CNN models for image-related tasks. For example, the work in [71] uses temporal convolutional networks for video summarization, where the task is formulated as predicting a binary label for each frame in a video. Our proposed model is based on the temporal convolution network proposed in [71], but we extend the model to perform user-specific video highlight detection.

5.2.2 Temporal-Adaptive Instance Normalization

Let $\mathbf{o}^i$ indicate the activations of i-th layer in the temporal convolution neural network (f_T) for the input video $\mathbf{v}$. We use C^i and T^i to denote the number of channels and temporal length of activation in that layer, respectively. We define the *Temporal-Adaptive Instance Normalization* (T-AIN), a conditional normalization layer for videos. T-AIN is inspired by the basic principles of Instance Normalization [86]. The activation is firstly normalized channel-wise along the *temporal* dimension (obtaining $\mathbf{o}^i_{norm}$), followed by a uniform modulation with affine transformation. Different from InstanceNorm [86], the affine parameters, scale and bias, in T-AIN are *not learnable* but inferred using external data (i.e., a user's history ($\mathcal{H}$) in our case) which is encoded to a vector m using another temporal convolution network (g_T). Thus, T-AIN is also *conditional* (on $\mathcal{H}$) in nature. The activation value from T-AIN

at location $c \in C^i$ and $t \in T^i$ is

$$\left(\frac{o^i_{c,t} - \mathrm{E}[o^i_c]}{\sqrt{\mathrm{Var}[o^i_c] + \epsilon}} \right) \gamma^i_c + \delta^i_c, \tag{5.2}$$

where $o^i_{c,t}$, $\mathrm{E}[o^i_c]$ and $\mathrm{Var}[o^i_c]$ are the activation before normalization, expectation and variance of the activation $\mathbf{o}^i$ in channel c, respectively. T-AIN computes the $\mathrm{E}[o^i_c]$ and $\mathrm{Var}[o^i_c]$ along the temporal dimension independently for each channel and every input sample (video) as:

$$\mathrm{E}[o^i_c] = \mu^i_c = \frac{1}{T^i}\Big(\sum_t o^i_{c,t}\Big), \tag{5.3}$$

$$\mathrm{Var}[o^i_c] = \mathrm{E}[(o^i_c - \mathrm{E}[o^i_c])^2] = \frac{1}{T^i}\sum_t \big(o^i_{c,t} - \mu^i_c\big)^2. \tag{5.4}$$

In Eq. 5.2, γ^i_c and δ^i_c are the modulation parameters in the T-AIN layer. We obtain γ^i_c and δ^i_c from the encoded vector m generated using the external data. T-AIN firstly scales each value along the temporal length in channel c of temporally normalized activations $\mathbf{o}^i_{norm}$ by γ^i_c and then shifts it by δ^i_c. Similar to InstanceNorm, these statistics vary across channel C^i. We provide details on how we compute these parameters when using user's history $\mathcal{H}$ as an external data in Sec. 5.2.3. In Fig. 5.2, we visualize the operations in a T-AIN layer.

T-AIN is related to the conditional batch normalization (CBN) [15] and the adaptive instance normalization (AIN) [31]. The main difference is that CBN and AIN work spatially, so they are appropriate for image-related tasks like style transfer. In contrast, T-AIN is designed to operate along time, which makes it suitable for video highlight detection and video understanding tasks in general.

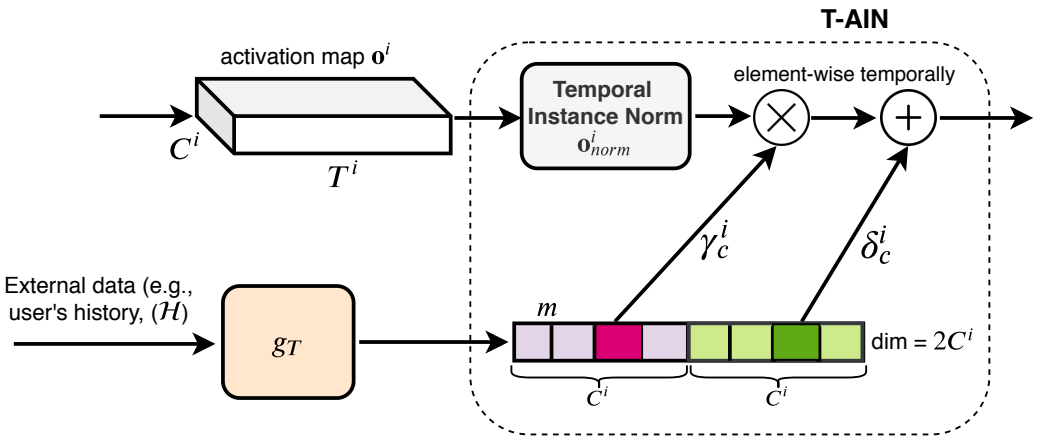

Figure 5.2: Overview of a temporal-adaptive instance normalization layer (T-AIN). For an input video $\mathbf{v}$, let $\mathbf{o}^i$ be the activation map with channel dimension C^i and temporal length T^i in the i-th layer of a temporal convolutional network f_T. Let g_T be another temporal convolutional network that encodes external data (e.g., user's history $\mathcal{H}$) into a vector representation m of dimension $2C^i$. T-AIN firstly temporally normalizes $\mathbf{o}^i$ in each channel to obtain $\mathbf{o}^i_{norm}$. It then uniformly scales and shifts $\mathbf{o}^i_{norm}$ in channel c (where $c \in C^i$) over time by γ^i_c and δ^i_c, respectively. The values of γ^i_c and δ^i_c are obtained from m. As can be seen, the main characteristics of T-AIN include temporal operation, no learnable parameters, and conditional on external data.

5.2.3 Adaptive Highlight Detector

The adaptive highlight detector G consists of two sub-networks: a highlight detection network H and a history encoder network M. H is responsible for scoring each frame in an input video to indicate whether or not it should be included in the highlight. The role of M is to firstly encode the user's history information and then guide H in a manner that the generated highlight is adapted to the user's history. Next, we discuss the sub-networks design and learning in detail.

Highlight Detection Network

The highlight detection network H is based on FCSN [71]. It is an encoder-decoder style architecture which is fully convolutional in nature. One advantage of this network is that it is not restricted to fixed-length input videos. It can handle videos with arbitrary lengths. Another advantage is that it is effective in capturing the long-term temporal dynamics among the frames of a video beneficial for video understanding tasks such as highlight detection.

H accepts a video $\mathbf{v}$ with feature representation of dimension $1 \times T \times D$, where T is number of frames in $\mathbf{v}$ and D is the dimension of each frame feature vector. It produces an output of dimension $1 \times T \times 2$ indicating non-highlight and highlight scores for the T frames.

The encoder (F_v) of H has a stack of seven convolutional blocks. The first five blocks (i.e., *conv-blk*1 to *conv-blk*5) consist of several temporal convolution followed by a ReLU and a temporal max pooling operations. The last two blocks (*conv*6 to *conv*7) have a temporal convolution, followed by a ReLU and a dropout layer. The encoder F_v gives two outputs: a feature map from its last layer and a skip connection from block *conv-blk*4.

The output of encoder F_v is fed to the decoder network (D_v). We introduce T-AIN layers at sites where these two outputs enter D_v. We obtain a feature map by applying a 1×1 convolution and a temporal fractionally-strided convolution operation (*deconv*1) to the output of first T-AIN, which is added with the feature map from a 1×1 convolution to the output of second T-AIN layer. Finally, we apply another fractionally-strided temporal convolution (*deconv*2) to obtain a final prediction of

shape $1 \times T \times 2$ denoting two scores (non-highlight or highlight) for each frame in the video. Fig. 5.3 (top) visualizes the architecture of H.

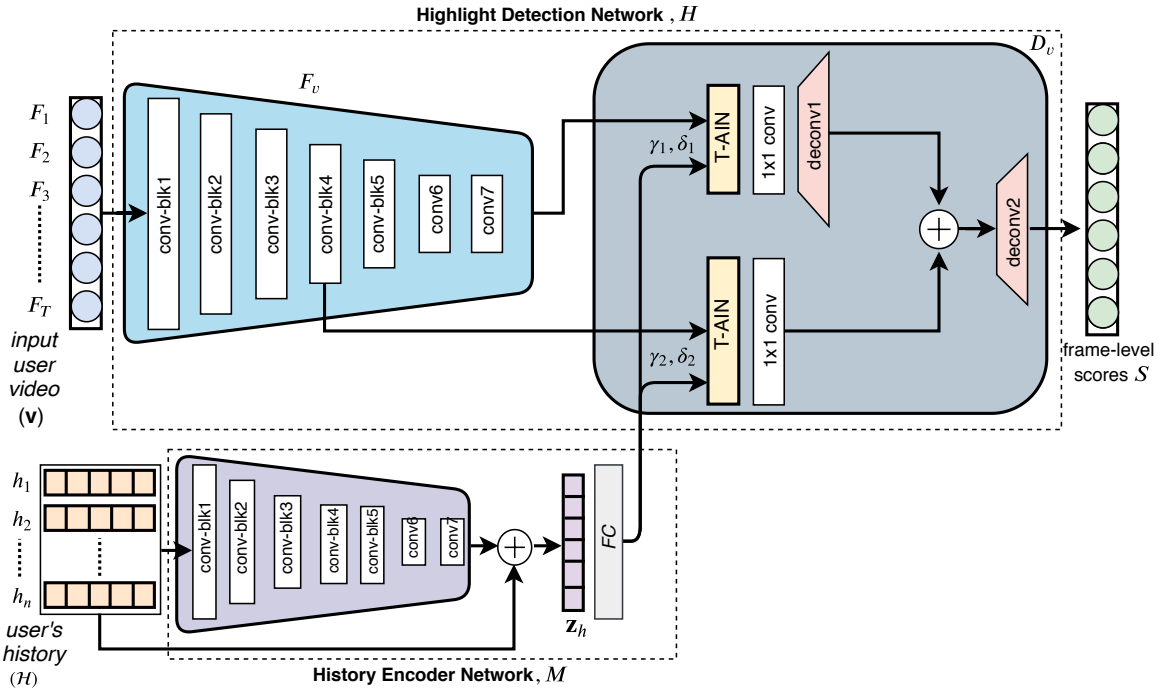

Figure 5.3: Overview of our proposed model, `Adaptive-H-FCSN`. The model consists of two sub-networks, a *highlight detection network* H and a *history encoder network* M. H is an encoder-decoder architecture that takes a frame-level vector feature representation of a user input video with T frames. It then generates scores (highlight vs. non-highlight) for each frame in the video while taking information from M. M takes vector feature representation of each element (i.e., highlights the user has previously created) in the user's history as an input and encodes it to a vector $\mathbf{z}_h$. This vector $\mathbf{z}_h$ is then simply fed to a fully connected layer FC to produce the affine parameters γ_j and δ_j in the j-th T-AIN layer of decoder D_v where $j = 1, 2$. This way the highlight detection for the input video is adapted to the user.

History Encoder Network

The history encoder network M is an integral piece of our framework. It acts as a guiding network that guides the highlight network H by adaptively computing the affine transformation parameters of its T-AIN layers. Using these affine parameters,

M modulates the activations in H to produce adaptive highlight predictions.

The configuration of this network is same as the encoder F_v in H with few changes towards the end. It performs an average temporal pooling to the output of convolution blocks, which is combined with a skip connection from the input that is then fed to a fully-connected layer. The skip connection involves an average pooling and a fully-connected operation to match dimensions.

The network accepts the user's history collection $\mathcal{H}$ of shape $1 \times n \times D$ as an input, where n is the number of elements/highlights in the user's history and D is the dimension of the vector representation of each element. In our implementation, we obtain a D-dimensional vector from a highlight using C3D [83]. Note that n varies for different users. After the combining stage, the network generates a latent code $\mathbf{z}_h$ which is a fixed-length user-specific vector.

Next, we forward $\mathbf{z}_h$ to a fully connected layer (FC) to decode the latent code $\mathbf{z}_h$ into a set of vectors (γ_j, δ_j) where $j = 1, 2$. The parameters γ_j and δ_j denote the scaling and bias parameters of j-th T-AIN layer in the decoder D_v of H, respectively. These affine parameters are uniformly applied at every temporal location in the feature map. This allows to incorporate the user's history information in H and adjust it in a way that the predicted highlight is adapted to the user's history. This way we obtain a user-specific highlight prediction for the input user video. Fig. 5.3 (bottom) presents the architecture of M.

With F_v, D_v and M, we can rewrite Eq. 5.1 as:

$$S = D_v(F_v(\mathbf{v}), M(\{h_1, ..., h_n\})). \tag{5.5}$$

By applying this design, we learn a generic video representation using F_v and extract

a user-specific latent representation with M. Finally, by injecting user-specific latent information to D_v through T-AIN layers, we allow the model to adapt the highlight detection to the user's history.

Note that we use temporal convolutions over the highlights in the user history. In addition, we also investigate a non-local model [90; 100] with self-attention instead of temporal convolutions for M. Here, the output of self-attention is firstly average pooled to produce a single vector and then fed to a fully-connected layer. We find that the non-local model performs slightly inferior to the temporal convolutions model. This is probably because the highlights in the history are ordered based on their time of creation in the dataset, so the temporal convolutions allow the history encoder M to capture some implicit temporal correlations among them.

5.2.4 Learning and Optimization

We train our adaptive highlight detector G using a cross-entropy loss. For an input video $\mathbf{v}$ with T frames and corresponding binary indicator ground-truth label vector (indicating whether a frame is non-highlight or highlight), we define a highlight detection loss $\mathcal{L}_{highlight}$ as:

$$\mathcal{L}_{highlight} = -\frac{1}{T}\sum_{t=1}^{T}\log\left(\frac{\exp(\lambda_{t,l_c})}{\sum_{c=1}^{2}\exp(\lambda_{t,c})}\right), \tag{5.6}$$

where $\lambda_{t,c}$ is the predicted score of t-th frame to be the c-th class (non-highlight or highlight) and λ_{t,l_c} is the score predicted for the ground-truth class.

The goal of our learning is to find optimal parameters $\Theta^*_{F_v}$, $\Theta^*_{D_v}$ and Θ^*_M in the encoder F_v, decoder D_v of the highlight detection network H, and the history encoder

network M, respectively. The learning objective can be expressed as:

$$\Theta^*_{F_v}, \Theta^*_{D_v}, \Theta^*_M = \underset{\Theta_{F_v}, \Theta_{D_v}, \Theta_M}{\arg\min} \mathcal{L}_{highlight}(F_v, D_v, M). \tag{5.7}$$

For brevity, we use `Adaptive-H-FCSN` to denote our adaptive highlight detection model learned using Eq. 5.7.

5.3 Experiments

In this section, we present the experimental details, results and analysis.

5.3.1 Dataset

We conduct experiments on the largest publicly available highlight detection dataset, PHD-GIFs [16]. It is also the only large-scale dataset that has user history information for highlight detection. The released dataset consists of 119,938 videos, 13,822 users and 222,015 annotations. The dataset has 11,972 users in training, 1,000 users in validation, and 850 users in testing. There is no overlap among users in these three subsets.

Apart from being large-scale, this dataset is also interesting because it contains user-specific highlight examples indicating what exactly a user is interested in when creating highlights. The ground-truth segment-level annotation comes from GIFs that a user creates (by extracting key moments) from YouTube videos. In this dataset, a user has GIFs from multiple videos where the last video of the user is considered for highlight prediction and the remaining ones are treated as examples in the user's history.

The dataset only provides YouTube video ID for the videos and not the original videos. So we need to download the original videos from YouTube to carry out the experiments. We were able to download 104, 828 videos and miss the remaining videos of the dataset since they are no longer available on YouTube. In the end, we are able to successfully process 7, 036 users for training, 782 users for validation and 727 users for testing. Note that code of previous methods on this dataset are not available (except pre-trained Video2GIF [28]), so we implement several strong baselines (see Sec. 5.3.3).

5.3.2 Setup and Implementation Details

Evaluation metrics: We use the mean Average Precision (mAP) as our evaluation metric. It measures the mean of the average precision of highlight detection calculated for every video in the testing dataset. Different from object detection where all the detections are accumulated from images to compute the average precision, highlight detection treats videos separately because it is not necessary a highlighted moment in a particular video is more interesting than non-highlighted moments in a different video [80]. This metric is commonly used to measure the quality of predictions in highlight detection [28; 16; 80; 95].

Feature representation: Following prior work [16], we extract C3D [83] (conv5) layer features and use it as feature representation in the model for the input videos and user's history. For an input video, we extract C3D-features at frame-level. For a highlight video in the user's history, we prepare a single vector representation by averaging its frame-level C3D features.

Training details: We train our models from scratch. All the models are trained with a constant learning rate of 0.0001. We use Adam [40] optimization algorithm for training the models. Note that we apply this training strategy in all our experiments including the other analysis (Sec. 5.3.5).

Since the dataset has users that create multiple GIFs for a video, we follow [16] to prepare a single ground truth for the video by taking their union.

Testing details: Given a new test user video and the user's history, we use our trained model to predict a highlight score for each frame which is then sent to the evaluation metrics to measure the quality of predicted highlight. We follow the evaluation protocol of previous work [28; 16] for fair comparison. Note that our model can handle variable length input videos and a variable number of history elements. We consider the full user's history while predicting highlights.

5.3.3 Baselines

We compare the proposed `Adaptive-H-FCSN` with the following strong baselines:

FCSN [71]: This network is the state of the art in video summarization which we adapt as our highlight detection network. FCSN has no instance normalization layers. We train and evaluate FCSN on the PHD-GIFs dataset.

Video2GIF [28]: This baseline is a state-of-the-art highlight detection model. We evaluate the publicly available pre-trained model.

FCSN-aggregate: In this baseline, we train FCSN [71] by directly combining the user history with the input video. More specifically, we firstly obtain a vector representation for the user history by averaging the features of elements in the history.

Next, we add this aggregated history with the feature representation of each frame in the input video.

H-FCSN: This baseline is a variant of highlight detection network H we presented in Sec. 5.2.3, where we replace the T-AIN layers in the decoder of H with the unconditional temporal instance normalization layers. We do not have the history encoder network M. This results in `Adaptive-H-FCSN` transformed to a generic highlight detection model with no adaptation to users.

H-FCSN-aggregate: Similar to FCSN-aggregate, we directly combine the user's history features with an input video features and learn H-FCSN. Different from H-FCSN, this is not a generic highlight detection model but a user-adaptive highlight detection model as we allow the model to leverage the user's history information in the training and inference.

5.3.4 Results and Comparison

In Table 5.1, we provide the experiment results (in terms of mAP %) of our final model `Adaptive-H-FCSN` along with the baselines and other alternative methods.

`Adaptive-H-FCSN` outperforms all the baselines. Results show that directly combining (i.e., FCSN-aggregate and H-FCSN-aggregate) history information with input video only slightly improves the highlight detection results in comparison to FCSN and H-FCSN that do not leverage users history information. However, we notice a significant performance gain for `Adaptive-H-FCSN` model. This result validates that directly combining user history information with the input video is a sub-optimal solution for user-adaptive highlight detection. Additionally, this result reveals that

Method	mAP (%)	User-adaptive
Random	12.27	✗
FCSN [71]	15.22	✗
Video2GIF [28]	14.75	✗
H-FCSN	15.04	✗
FCSN-aggregate	15.62	✓
H-FCSN-aggregate	15.73	✓
`Adaptive-H-FCSN-attn`	16.37	✓
`Adaptive-H-FCSN`	**16.73**	✓

Table 5.1: Performance (mAP%) comparison between `Adaptive-H-FCSN` and other approaches. We compare with both non-adaptive and adaptive highlight detection methods. Our method `Adaptive-H-FCSN` outperforms the other alternative methods. We also compare with `Adaptive-H-FCSN-attn` that uses self-attention in the history encoder (see Sec. 5.2.3). Note that all the listed methods use C3D feature representation.

proposed T-AIN layer plays a critical role in producing more accurate and user-specific highlight detection. It is also noteworthy that we obtain a lower performance (nearly 1%) for Video2GIF [28] than reported in PHD-GIFs [16] which implies that our test set is more challenging.

Fig. 5.4 shows some qualitative examples for the generic baseline model (H-FCSN) and our proposed adaptive highlight detection model (`Adaptive-H-FCSN`). We can see that our model successfully captures the information in the user's history and produces a video highlight that is adapted to the user.

Figure 5.4: Qualitative examples for different methods. We show examples of the generic highlight detection model (H-FCSN) and our user-adaptive model (`Adaptive-H-FCSN`) on four videos. For each video, we show the user's history (multiple GIFs) and few sampled frames from the highlight predictions of the two models. Based on the user's history, we find that in (a) the user has interest in animals; (b) the user is interested in faces that dominate a scene; (c) the user is inclined to highlight goal scoring scenes; and (d) the user focuses on cooking. These visualizations indicate that adaptation to the user's interest is important for a meaningful and accurate highlights. Compared with H-FCSN, the prediction of `Adaptive-H-FCSN` is more consistent with the user's history.

5.3.5 Analysis

Effect of affine parameters

We analyze the importance of affine parameters γ_c^i and δ_c^i (Eq. 5.2) for adaptive highlight detection. In Table 5.2, we report the highlight detection performance for different possible choices of these parameters. We find that when these parameters are adaptively computed ($\gamma_c^i{=}\gamma_c^h$, $\delta_c^i{=}\delta_c^h$) from another network capturing the user history information (i.e., `Adaptive-H-FCSN`) significantly boosts the highlight detec-

Method	γ_c^i=1, δ_c^i=0	γ_c^i=γ_c^*, δ_c^i=δ_c^*	γ_c^i=γ_c^h, δ_c^i=δ_c^h
H-FCSN	14.64	15.04	-
H-FCSN-aggregate	14.87	15.73	-
`Adaptive-H-FCSN`	-	-	**16.73**

Table 5.2: Impact of affine parameters on highlight detection. Here we show the performance (mAP%) for different choices of affine parameters γ_c^i and δ_c^i in Eq. 5.2.

tion performance as opposed to cases when it is directly learned (γ_c^i=γ_c^*, δ_c^i=δ_c^*) and set to a fixed value (γ_c^i=1, δ_c^i=0) in the main highlight detection network. Thus, the proposed T-AIN layer is key to obtain user-adaptive highlights.

Effect of user's history size

We perform an additional study to analyze how sensitive our model is to the length of a user's history (i.e., numbers of highlights previously created). We restrict the number of history elements for users in the training. That is, we consider only h highlight videos from the user's history in training. During testing, we consider the user's full history.

Table 5.3 shows the results of various methods as a function of number of elements ($h = 0, 1, 5, n$) in user's history. We observe that `Adaptive-H-FCSN` outperforms generic highlight model (H-FCSN) even when there is a single highlight in the user's history. We notice the performance of `Adaptive-H-FCSN` gradually improves when we increase the number of history elements, whereas H-FCSN-aggregate doesn't show a similar performance trend. It achieves the best results when we utilize a user's full history (i.e., h=n).

History size ($\mathcal{H}$)	$h = 0$	$h = 1$	$h = 5$	$h = n$
H-FCSN	15.04	-	-	-
H-FCSN-aggregate	-	15.62	15.04	15.73
`Adaptive-H-FCSN`	-	15.57	15.69	**16.73**

Table 5.3: Impact of an user's history size (i.e., number of history elements/highlights) on different methods. Here we vary the history size h as 0 (no history), 1, 5, and n (full history). The performance of our model improves with the increase in history size.

5.3.6 Application to Video Summarization

Video summarization is closely related to highlight detection. Highlight detection aims at extracting interesting moments and events of a video, while video summarization aims to generate a concise synopsis of a video. Popular datasets in summarization are very small [70], making learning and optimization challenging. We argue that pre-training using a large-scale video data from a related task, such as PHD-GIFs [16] in highlight detection, could tremendously help video summarization models. In video summarization, this idea remains unexplored. In order to validate our notion and compare with recent state-of-the-art in [92], we select the SumMe dataset [26] which has only 25 videos.

We evaluate our trained H-FCSN (i.e., the generic highlight detection model we proposed in Sec. 5.3.3) directly on SumMe. In Table 5.4, we compare the performance of our H-FCSN (trained on the PHD-GIFs [16] dataset) on SumMe with state-of-the-art supervised video summarization methods. Following prior work [92], we randomly select 20% of data in SumMe for testing. We repeat this experiment five times (as

in [92]) and report the average performance. Surprisingly, even though we *do not* train on SumMe, our model achieves state-of-the-art summarization performance, outperforming contemporary supervised models. Therefore, we believe that future research in video summarization should consider pretraining their model on a large-scale video data from a related task such as highlight detection. We envision that this way we can simultaneously make progress in both highlight detection and video summarization.

Method	F-score (%)
Interesting [26]	39.4
Submodularity [27]	39.7
DPP-LSTM [103]	38.6
GAN$_{sup}$ [58]	41.7
DR-DSN$_{sup}$ [110]	42.1
S^{2}N [92]	43.3
Ours (H-FCSN)	**44.4**

Table 5.4: Performance comparison in term of F-score (%) on SumMe. Note that unlike other methods, we do not train on SumMe rather directly test our trained (using PHD-GIFs) model for summarization. Results of other methods are taken from [92].

5.4 Summary

We have proposed a simple yet novel framework `Adaptive-H-FCSN` for adaptive highlight detection using the user history which has received less attention in the

literature. Different from commonly applied ranking-based models, we introduced a convolutional model for highlight detection that is computationally efficient as it can process an entire video of any length at once and also does not require expensive shot detection computation. We proposed temporal-adaptive normalization (T-AIN) that has affine parameters which is adaptively computed using the user history information. The proposed T-AIN leads to high-performing and user-specific highlight detection. Our empirical results on a large-scale dataset indicate that the proposed framework outperforms alternative approaches. Lastly, we further demonstrate an application of our learned model in video summarization where the learning is currently limited to small datasets.

Chapter 6

Learning to Generate Dynamic Video Thumbnail using Sentences

6.1 Chapter Overview and Introduction

In this chapter, we propose a sentence-guided temporal modulation for dynamic video thumbnail generation. In addition, we introduce self-supervision through an auxiliary task in our framework that not only improves the performance of the model but also yields an unsupervised approach for dynamic video thumbnail generation.

With the massive rise in videos available online, video thumbnails play a vital role in defining the browsing and searching experience of users. A video thumbnail shows viewers a quick and condensed preview of the entire content in the video [78; 53; 99]. Viewers often decide whether to watch or skip the video based on its thumbnail [13; 78]. Given its importance, there is increasing interest in how to create attractive and expressive thumbnails.

Most of the existing approaches [17; 29; 37; 56; 78; 91] focus on static thumbnail generation, where a thumbnail is generated solely based on the input video. These static thumbnail generation methods overlook rich semantic information such as user query sentence which is usually provided in searching a video. Static thumbnails are not tailored to each viewer's unique interest and may not provide the best online videos browsing experience. Some approaches [49; 53] consider the user search query for video thumbnail generation. But they either limit the thumbnail to a single keyframe or confine queries to a single word or a short phrase [99]. In this chapter, we study the recently proposed challenging problem of sentence specified dynamic video thumbnail generation (DVTG) [99]. Given an input video and a user query expressed as a free form natural language sentence, the goal of DVTG is to generate a video thumbnail that semantically corresponds to the sentence while giving a concise preview of the video. Figure 6.1 shows the difference between static thumbnail generation and DVTG. Despite many potential real world applications (e.g., video search), there is limited work on the DVTG problem.

The existing state-of-the-art method [99] for the DVTG task achieves promising results but has some limitations. First, it heavily relies on fine-grained modeling of semantic relationship within the user query sentence and clips or segments of the video [99]. It does not consider the overall guiding and modulating role that a user query sentence can play in temporally correlating video clips over time. Intuitively, global sentence semantics can serve as a reference in determining and associating sentence-specific video segments over time. Second, the method uses recurrent models (BiGRU [11] and pointer networks [89]) that are hard to parallelize as they inherently perform

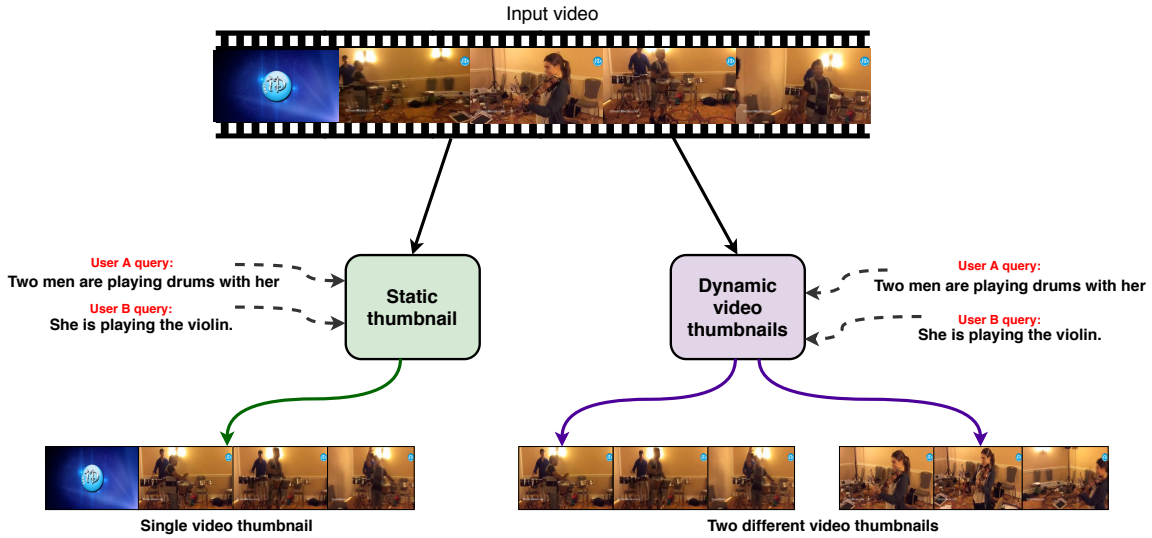

Figure 6.1: Illustration of the difference between static video thumbnails and sentence specified dynamic video thumbnails. The latter considers the user query sentence when creating the thumbnail.

sequential computation.

In this chapter, we propose a sentence-guided temporal modulation (SGTM) mechanism for the DVTG task. We use a self-attention network [90] for encoding the user query sentence and adapt a fully convolutional temporal network [71] for the thumbnail generation of an input video. The SGTM mechanism leverages the semantic information from the user query sentence to modulate the normalized temporal activations in the video thumbnail generation network. We also introduce a small auxiliary network to further ensure that the generated video thumbnail semantically corresponds to the given user sentence. Our framework is computationally efficient as its computations are easily parallelizable on GPU architectures. It is also noteworthy that our framework is free from sophisticated multimodal feature fusion (unlike prior method [99]), making it computationally less complex. We also propose an unsupervised extension of our method that does not need ground-truth video thumbnails

during training.

In summary, our contributions in this chapter are as follows. (1) We propose a sentence-guided temporal modulation (SGTM) mechanism for sentence specified dynamic video thumbnail generation (DVTG). Our method dynamically modulates the temporal activations in video thumbnail generation network using the semantic information of the query sentence. (2) We propose a computationally efficient and simple framework for the DVTG task that offers better parallelization and is free from complex multimodal feature fusion. (3) We propose both supervised and unsupervised version of our method. (4) We conduct extensive experiments and analysis on a large-scale dataset to evaluate our framework.

6.2 Our Approach

The input to DVTG consists of a video V and a user query sentence S. We denote video by $V = \{v_c\}_{c=1}^{C}$, where v_c is the feature representation of c-th video clip and C is the total number of clips in the video. Similarly, we represent the user query sentence by $S = \{x_n\}_{n=1}^{N}$, where x_n denotes the word embedding of n-th word and N denotes the total number of words in the sentence. The goal of DVTG is to identify a set of video clips (which may not be consecutive) from V that provide a good preview of the original video V and are semantically consistent with the sentence S.

Given V and S, our goal is to learn a mapping function $F(V, S) \in \mathbb{R}^{1\times C\times 2}$, where the output of $F(V, S)$ indicate the scores of whether or not a video clip should be included in the video thumbnail. We call the function F the sentence-guided video thumbnail generation model.

6.2.1 Sentence-Guided Video Thumbnail Generation Model

We propose the sentence-guided video thumbnail generation model F which consists of three sub-networks, namely a video thumbnail generation network (T), a self-attention sentence encoder network (S_{enc}) and an auxiliary network (T_{aux}). A sentence-guided temporal modulation (SGTM) mechanism is proposed to modulate certain activations in T using the output from S_{enc}. Figure 6.2 shows an overview of our proposed model. In the following, we discuss our model in detail.

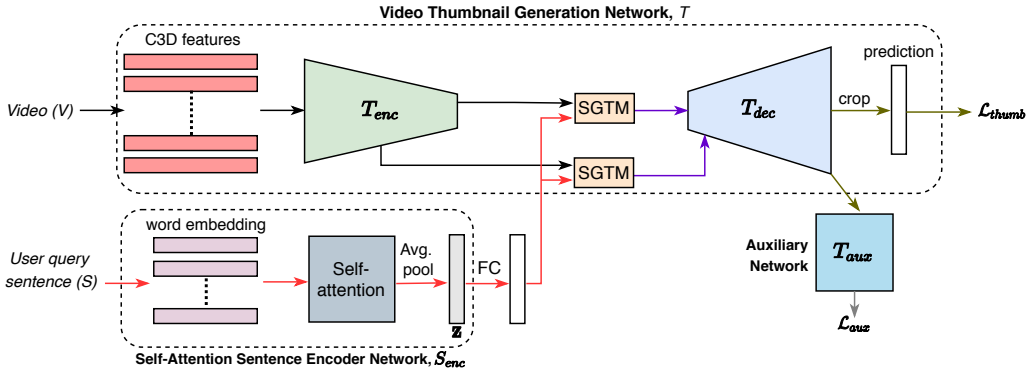

Figure 6.2: An overview of our sentence-guided video thumbnail generation model. The model consists of a video thumbnail generation network (T), a self-attention sentence encoder network (S_{enc}), and an auxiliary network (T_{aux}). A sentence-guided temporal modulation (SGTM) mechanism is introduced to allow interaction between T and S_{enc}. The network T consists of an encoder T_{enc} and a decoder T_{dec}. T takes features of video clips in an input video V and predicts whether or not each clip belongs to the video thumbnail. S_{enc} encodes the word-level embedding of user query sentence (S) to a vector $\mathbf{z}$ which is then used by SGTM to modulate the temporal activations from the encoder of T (i.e., T_{enc}) to determine sentence-specific video content over time. The role of T_{aux} is to reconstruct $\mathbf{z}$ to further ensure that the generated video thumbnail aligns well with S. We use two losses for learning: a thumbnail generation loss $\mathcal{L}_{thumb}$ on the prediction of T_{dec} and an auxiliary loss $\mathcal{L}_{aux}$ on the output of T_{aux}.

Video Thumbnail Generation Network

The video thumbnail generation network T is an encoder-decoder style temporal convolution network. A temporal convolution network mainly performs 1D operations (e.g., 1D convolution, 1D pooling) over time. For instance, given a video with frame-level feature representations, this network can operate over frames enabling it to capture the temporal dependencies among the frames. In this chapter, we adapt FCSN [71] which is a form of encoder-decoder based temporal convolution network designed for video summarization.

The input to T is a sequence of clip-level feature representations of video V, i.e., $V \in \mathbb{R}^{1\times C\times D_c}$ where C is the total number of clips and D_c is the dimension of the feature representation of each video clip. The output of T is of dimension $1 \times C \times 2$ which denotes the scores of each clip being a part of the thumbnail or not.

T consists of an encoder T_{enc} and a decoder T_{dec}. T_{enc} has seven convolutional blocks. Each of the first five convolution blocks ($conv1$ to $conv5$) consists of multiple temporal convolution and ReLU operations, with a max pooling at the end. The last two blocks ($fc6$ and $fc7$) has a temporal convolution, a ReLU and a dropout operation. T_{enc} produces two feature maps as outputs, one from the last layer and another one as a skip connection from $conv4$ block. The outputs of T_{enc} is fed to T_{dec} as inputs. The first input to T_{dec} goes through a 1×1 convolution and a temporally fractionally-strided convolution ($deconv1$) which is then combined with the second input after applying a 1×1 convolution to it. Lastly, it has another temporally fractionally-strided convolution ($deconv2$) and a crop operation so as to obtain the final prediction of dimension $1 \times C \times 2$.

Self-Attention Sentence Encoder Network

The self-attention sentence encoder network (S_{enc}) is responsible for encoding the user query sentence S to a fixed length vector $\mathbf{z}$.

Self-attention [65; 88] has been shown to be a powerful technique in natural language processing. We apply the non-local model [90; 100] with self-attention mechanism in S_{enc}. This enables S_{enc} to effectively model relationships between different words in the sentence.

We represent each word in the sentence using its word embedding vector. The sentence S can be written as $S \in \mathbb{R}^{D_w \times N}$, where N is the number of words in the sentence and D_w is the dimension of word embedding. The attention between two words can be computed as:

$$\Phi_{j,i} = \frac{\exp(A_{ij})}{\sum_{i=1}^{N} \exp(A_{ij})}, \quad \text{where} \quad A_{ij} = f_1(x_i)^T f_2(x_j), \tag{6.1}$$

Here f_1 and f_2 represent two distinct feature spaces, i.e., $f_1(x_i) = W_{f_1} x_i$ and $f_2(x_j) = W_{f_2} x_j$. The attention score $\Phi_{j,i}$ denotes the extent with which i-th word is related to j-th word. The output of self-attention is $\Lambda = (\Lambda_1, \Lambda_2, ..., \Lambda_N) \in \mathbb{R}^{D_w \times N}$, where

$$\Lambda_j = \sum_{i=1}^{N} \Phi_{j,i} h(x_i), \quad \text{where} \quad h(x_i) = W_h x_i. \tag{6.2}$$

Note that $W_{f_1} \in \mathbb{R}^{d \times D_w}$, $W_{f_2} \in \mathbb{R}^{d \times D_w}$ and $W_h \in \mathbb{R}^{D_w \times D_w}$ are the learnable weights implemented using 1×1 convolutions. In our experiments, we set $d = D_w/8$. Lastly, we apply average pooling on the resultant self-attended features Λ to obtain the vector $\mathbf{z} \in \mathbb{R}^{D_w}$ representing the whole sentence S.

Sentence-Guided Temporal Modulation

In order to generate a video thumbnail that corresponds to a user sentence, it is necessary to establish the relationships between the clips of video and the sentence. The rich semantics in the sentence provides a strong signal to temporally associate it with the video clips. Motivated by this, we propose the sentence-guided temporal modulation (SGTM) mechanism. In practice, SGTM is similar to temporal-adaptive instance normalization [69] that extends adaptive instance normalization [31] (initially designed for images) to videos. SGTM plays a key role in our proposed model. It allows the interaction between the video thumbnail generation network T (Sec. 6.2.1) and the self-attention sentence encoder network S_{enc} (Sec. 6.2.1) so as to generate a video thumbnail that closely relates to the given user query sentence. SGTM uses the semantic information in the sentence to guide and temporally modulate the features of the input video fed to the video thumbnail generation network T.

Let $\mathbf{z}$ be the sentence vector representation from S_{enc} and $A \in \mathbb{R}^{1\times M\times C}$ be the activations from one specific layer in the temporal convolution network T, where M is the temporal length of the activation and C is the number of channels. We firstly forward $\mathbf{z}$ to a fully-connected layer (FC) to generate a vector of length $2C$. We use this generated vector to obtain two modulation vectors $\alpha \in \mathbb{R}^C$ and $\beta \in \mathbb{R}^C$. Next, we use these modulation vectors to modulate the each channel of activation map A. We can express the modulated activation map at $m \in M$ and $c \in C$ as $\hat{A}_{m,c}$ which is computed as:

$$\hat{A}_{m,c} = \alpha_c \cdot \frac{A_{m,c} - \mu_c(A)}{\sigma_c(A)} + \beta_c, \tag{6.3}$$

where μ_c and σ_c are the channel-wise mean and standard deviation calculated in-

dependently for each input video along the temporal length of the activation map A.

From Eq. 6.3, we can infer that a normalized temporal activation map with zero mean and unit variance in each channel is subjected to an affine transformation (scale and shift) whose values are predicted using the sentence representation $\mathbf{z}$. Note that the affine transformation is temporally invariant and there are no learnable parameters involved in this mechanism. By using this translation-based design, our goal is to allow the sentence semantics information to modulate each temporal feature map of the input video.

In our model, we apply SGTM to the two output activations of the encoder T_{enc} (see Fig. 6.2). We use the sentence representation $\mathbf{z}$ from S_{enc} to produce two sets of affine parameter vectors (α_i, β_i where $i = 1, 2$) using a FC layer. The parameters α_i and β_i correspond to the i-th SGTM.

Auxiliary Network

To further ensure that the generated thumbnail aligns well with the query sentence, we introduce an auxiliary network (T_{aux}) next to the decoder T_{dec} of T (Sec. 6.2.1). It is a small network whose input is the output of *deconv*2 layer in T_{dec} which is forwarded to a 1×1 convolution and an average pooling operation so as to reconstruct the user sentence vector representation $\mathbf{z}$ learned by S_{enc} (Sec. 6.2.1).

6.2.2 Learning and Optimization

Our learning objective includes a thumbnail loss and an auxiliary loss.

Thumbnail loss: For an input video V with C clips and the ground-truth binary indicator label vector denoting whether a clip belongs to thumbnail or not, we define a cross-entropy loss $\mathcal{L}_{thumb}$ on the prediction of the video thumbnail generation network T as:

$$\mathcal{L}_{thumb} = -\frac{1}{C}\sum_{c=1}^{C}\log\left(\frac{\exp(\delta_{c,l_c})}{\sum_{j=1}^{2}\exp(\delta_{c,j})}\right), \tag{6.4}$$

where $\delta_{c,j}$ is the predicted score of c-th video clip to be labeled as j-th class (non-thumbnail or thumbnail) and δ_{c,l_c} is the score for the ground-truth class l_c for the c-th video clip.

Auxiliary loss: This loss aims to minimize the difference between the sentence representation $\mathbf{z}$ from S_{enc} and the reconstructed sentence representation $\hat{\mathbf{z}}$ from T_{aux}. We define the reconstruction loss $\mathcal{L}_{aux}$ as:

$$\mathcal{L}_{aux} = ||\mathbf{z} - \hat{\mathbf{z}}||^2, \tag{6.5}$$

where $\mathbf{z}, \hat{\mathbf{z}} \in \mathbb{R}^{D_w}$ and $||\cdot||$ denotes the L_2 norm.

Final loss: We define the final loss $\mathcal{L}_{final}$ as:

$$\mathcal{L}_{final} = \mathcal{L}_{thumb} + \mathcal{L}_{aux}. \tag{6.6}$$

The aim of the learning is to find the optimal parameters Θ^*_T, $\Theta^*_{S_{enc}}$ and $\Theta^*_{T_{aux}}$ of the networks T, S_{enc} and T_{aux}, respectively. We can express the learning objective as follows:

$$\Theta^*_T, \Theta^*_{S_{enc}}, \Theta^*_{T_{aux}} = \underset{\Theta_T, \Theta_{S_{enc}}, \Theta_{T_{aux}}}{\arg\min} \mathcal{L}_{final}(T, S_{enc}, T_{aux}). \tag{6.7}$$

For simplicity, we denote our proposed sentence-guided dynamic video thumbnail generation model (Sec. 6.2.1) learned from Eq. 6.7 by `Guided-DVTG`.

In summary, `Guided-DVTG` captures the global information of the video through its video thumbnail generation network (Sec. 6.2.1) which is crucial to generate thumbnails that provide overall content preview. It also has the ability to dynamically modulate its prediction using the user sentence via the sentence-guided temporal modulation (Sec. 6.2.1). Moreover, it further ensures semantic correspondence with the sentence using the auxiliary network (Sec. 6.2.1). As a result, `Guided-DVTG` can produce dynamic video thumbnails that provide a quick preview of the video while satisfying the user query sentence.

6.3 Experiments

6.3.1 Setup

Dataset: We conduct experiments on the sentence specified dynamic video thumbnail generation dataset by Yuan *et al.*[99]. This dataset is based on the ActivityNet Captions dataset [42]. It has $10,204$ video-sentence pairs where each pair is labeled with 4 video thumbnail annotations. The thumbnail annotation for each video is at clip-level where each clip is of 2 seconds and with no more than 5 clips from the video included in the final video thumbnail. 70% of the dataset is used for training, 15% for validation and the remaining 15% for testing.

Feature representation: We follow prior work [99] for video and word embedding representation. We evenly split every video in the dataset into 2 second clips and

represent each clip with the C3D features [83] provided by the ActivityNet Challenge 2016. For each word, we obtain a 300 dimensional word embedding using Glove [67].

Training details: We train all our models from scratch with a constant learning rate 0.0001 using the Adam optimizer [40]. During training, for a video-sentence pair, we find the most consistent video thumbnail annotation among the 4 thumbnail annotations and treat it as the ground-truth annotation. However, in testing, we evaluate the predicted video thumbnail by comparing it against all the 4 thumbnail annotations. Note that similar process is followed by previous work [99].

Evaluation metrics: Following prior work [99], we use the F1 and IoU scores to measure the performance of our methods. These metrics measure the agreement between the generated thumbnail and the ground-truth thumbnail annotations. A higher value is desirable on these metrics.

6.3.2 Main Results and Comparisons

In addition to comparing with prior state-of-the-art video thumbnail generation methods, we also define several strong baselines that are as follows:

FCSN [71]: This is a state-of-the-art model in video summarization task. We extend it to create our video thumbnail generation network T (see Sec. 6.2.1). We directly train and evaluate FCSN on the dataset in this chapter. Note that this is a generic video thumbnail model that does not consider the user query sentence.

IN-FCSN: This baseline is a variant of video thumbnail generation network T where we replace the proposed SGTM mechanism (see Sec. 6.2.1) with the temporal instance normalization layer [85] with learnable affine transformation parameters. Note that

we do not have the networks S_{enc} and T_{aux} in this model. This results in another generic thumbnail model.

IN-FCSN-concat: We obtain this baseline when we train video thumbnail generation network T by concatenating the user sentence vector representation (obtained by averaging the words embedding) with the video-clip features of the input video. We again replace SGTM mechanism with temporal instance normalization layer with learnable affine transformation parameters. This results in a dynamic video thumbnail generation model as it incorporates user sentence information in the model.

In Table 6.1, we compare our final model `Guided-DVTG` with the prior and baseline methods. We outperform the baselines and other alternative methods except GTP [99]. Unlike GTP, we do not use sophisticated multimodal feature fusion. Moreover, GTP uses recurrent models that perform sequential computation within training samples which prevents parallelization, whereas our model is completely non-recurrent that allows much more parallelization. Figure 6.3 shows example video thumbnails generated by our `Guided-DVTG` model.

6.3.3 Analysis

Role of Modulation Parameters: We analyze the importance of modulation parameters α_c and β_c (Eq. 6.3) in the SGTM mechanism (Sec. 6.2.1). Table 6.2 compares the performance for different possible solutions of these parameters. We find that when these parameters are predicted (α_c=α_c^s, β_c=β_c^s) from another network using the user query sentence (i.e., in `Guided-DVTG`), the model achieves much better performance as compared to cases when they are set to fixed values (α_c=1, β_c=0)

Method	F1	IoU
Random	0.3604	0.2379
RankNet [28]	0.4013	0.2770
VSEM [53]	0.4386	0.3098
QARE [87]	0.4285	0.2986
CTRL [20]	0.4303	0.3084
ACRN [50]	0.4456	0.3271
GTP [99]	0.5285	0.3933
FCSN [71] (ours)	0.4295	0.3101
IN-FCSN (ours)	0.4426	0.3140
IN-FCSN-concat (ours)	0.4286	0.3084
`Guided-DVTG` (ours)	0.4758	0.3405

Table 6.1: Performance comparison (in terms of F1 and IoU) between `Guided-DVTG` and other alternative methods. Results of previous methods is taken from [99]. Best and second best methods are highlighted in gray and cyan, respectively.

or directly learned (α_c=α_c^*, β_c=β_c^*) in the main video thumbnail generation network (Sec. 6.2.1). Therefore, the proposed SGTM is key to dynamic video thumbnail generation.

Impact of Auxiliary Network: We study the impact of auxiliary network (Sec. 6.2.1) and the loss $\mathcal{L}_{aux}$ (Sec. 6.2.2) associated with it in learning the thumbnail model. In order to verify their contribution, we remove them from our final model `Guided-DVTG` and perform learning. We call the learned model `Guided-DVTG-NA` and compare the performance in Table 6.3(a). We notice a drop in performance which

Figure 6.3: Example qualitative results produced by `Guided-DVTG`. The gray, green and orange bars indicate the video length, ground truth and thumbnail predictions, respectively.

Method	$\alpha_c = 1$, β_c=0	α_c=α_c^*, β_c=β_c^*	α_c=α_c^s, β_c=β_c^s
IN-FCSN	0.4062 (0.2904)	0.4426 (0.3140)	-
IN-FCSN-concat	0.4480 (0.3192)	0.4286 (0.3084)	-
`Guided-DVTG`	-	-	**0.4758** (**0.3405**)

Table 6.2: Impact of modulation parameters on video thumbnail generation. Here we indicate F1 and IoU (in bracket) for different solutions of parameters α_c and β_c in Eq. 6.3.

highlights the importance of the auxiliary network and its loss in our `Guided-DVTG` model.

Unsupervised Guided-DVTG: We develop an unsupervised variant of our `Guided-DVTG`

model. When we remove the supervised loss $\mathcal{L}_{thumb}$ (Sec. 6.2.2 and Eq. 6.6) and perform learning only using the auxiliary loss $\mathcal{L}_{aux}$ which is completely unsupervised, we obtain the unsupervised version of our model to which we call `Guided-DVTG`$_{unsup}$. In Table 6.3(b), we compare its performance with the state-of-the-art unsupervised method, BeautThumb [78], when evaluated on the dataset in this chapter. Our model `Guided-DVTG`$_{unsup}$ significantly outperforms BeautThumb [78]. This result is very appealing since gathering labeled video thumbnail data is extremely expensive.

Method	F1	IoU
`Guided-DVTG`	**0.4758**	**0.3405**
`Guided-DVTG-NA`	0.4644	0.3296

(a)

Method	F1	IoU
Random	0.3604	0.2379
BeautThumb [78]	0.3837	0.2544
`Guided-DVTG`$_{unsup}$	**0.4222**	**0.2835**

(b)

Table 6.3: (a) Impact of auxiliary network and its loss. (b) Performance comparison of unsupervised methods. Result of BeautThumb [78] is taken from [99].

6.4 Summary

In this chapter, we have proposed a simple yet effective framework for the DVTG task. At the core of our framework is the proposed SGTM mechanism that modulates the normalized temporal activations in the video thumbnail generation network to effectively correlate sentence-specific video clips over time. Instead of applying recurrent neural architectures, we propose a non-recurrent solution that offers much more parallelization on GPU hardware. Our proposed framework does not involve complex multimodal feature fusion commonly used in vision-language tasks such as

DVTG. The experimental results and analysis on a large-scale dataset demonstrate that our proposed method achieves superior or competitive performance against the state-of-the-art methods.

Chapter 7

Conclusion

We have witnessed rapid growth in video data over the past few years. There is a pressing need for developing automated tools for video abstraction so as to support efficient video browsing, editing, searching, and categorizing. Video abstraction can provide viewers a quick perspective of the content of videos without watching them. Consequently, it can make the enormous video data more accessible and informative for users. In this thesis, we have developed deep learning models and techniques that push the frontier in video abstraction.

In Chapter 3, we proposed fully convolutional sequence networks for video summarization. Different from the state-of-the-art recurrent models for video summarization that precludes parallelization withing training examples due to sequential operations, our models offered better GPU parallelization and also achieved superior performance.

In Chapter 4, we introduced a novel formulation of learning video summarization using unpaired training data. The unpaired data consists of a set of raw videos and a set of video summaries, where there exists no correspondence information between

these two sets. We argued that the proposed unpaired training data is much easier to obtain than the standard paired training data in supervised learning. With this unpaired training data, we learned a model through an adversarial framework that produced promising results on the benchmark video summarization datasets.

In Chapter 5, we developed a high-performing adaptive video highlight detection model that learns to produce user-specific and personalized highlights by leveraging a user's previously created visual highlights history from different videos. We proposed a conditional temporal-adaptive instance normalization layer to induce personalization signal from the user history in the model.

Finally, in Chapter 6, we presented a new architecture for sentence-guided dynamic video thumbnail generation. We introduced sentence-guided temporal modulation that modulates the activations in the video thumbnail generation network based on a natural language textual query. In addition, we explored self-supervision through an auxiliary task that not only improved the performance of the model but also resulted in an unsupervised dynamic video thumbnail generation method.

We hope that this thesis can help inspire future research in video abstraction. There are several interesting and fascinating avenues for future research in this area. For example, currently vision and text modality, to an extent, have been well studied for model learning in video abstraction. However, the usage of audio modality that could also provide a strong learning signal is often ignored. We should aim to leverage the multimodal nature of videos for video abstraction in the future. Another exciting direction is to extend the proposed models to the rapidly growing 360° videos. It would be also exciting direction to explore video abstraction on compressed videos.

There is evidence in the computer vision literature (e.g., [94]) suggesting that operating directly on compressed videos can be advantageous. The video compression algorithms (e.g., MPEG-4) already remove redundant information from a video, making useful content of the video more prominent. However, existing research decompresses a video to raw RGB frames in which usually most of the frames could possibly be repeating and redundant. Our intuition is that this superfluous information makes harder for models to find meaningful information for video abstraction. Lastly, although we emphasized on video abstraction in this thesis, we wish that our proposed models can be reused in other video understanding tasks (e.g., video classification, action recognition, and video caption generation) that share very close connections with it. We envision that this would pave the way for comprehensive video understanding and ultimately enable machines to understand videos like humans.

Bibliography

[1] Cisco visual networking index: Forecast and methodology, 2016-2021. `https://www.cisco.com/`.

[2] Open video project. `https://open-video.org/`.

[3] L. Agnihotri, J. Kender, N. Dimitrova, and J. Zimmerman. Framework for personalized multimedia summarization. In *ACM SIGMM International Workshop on Multimedia Information Retrieval*, pages 81–88, 2005.

[4] A. Almahairi, S. Rajeswar, A. Sordoni, P. Bachman, and A. Courville. Augmented cyclegan: Learning many-to-many mappings from unpaired data. In *International Conference on Machine Learning*, pages 195–204, 2018.

[5] L. Anne Hendricks, O. Wang, E. Shechtman, J. Sivic, T. Darrell, and B. Russell. Localizing moments in video with natural language. In *IEEE International Conference on Computer Vision*, pages 5803–5812, 2017.

[6] N. Babaguchi, K. Ohara, and T. Ogura. Learning personal preference from viewer's operations for browsing and its application to baseball video retrieval and summarization. *IEEE Transactions on Multimedia*, 9(5):1016–1025, 2007.

[7] S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv:1803.01271*, 2018.

[8] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *International Conference on Computational Statistics*, pages 177–186, 2010.

[9] S. Cai, W. Zuo, L. S. Davis, and L. Zhang. Weakly-supervised video summarization using variational encoder-decoder and web prior. In *European Conference on Computer Vision*, pages 193–210, 2018.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(04):834–848, 2018.

[11] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014.

[12] W.-S. Chu, Y. Song, and A. Jaimes. Video co-summarization: Video summarization by visual co-occurrence. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3584–3592, 2015.

[13] S. J. Cunningham and D. M. Nichols. How people find videos. In *ACM/IEEE Joint Conference on Digital Libraries*, pages 201–210, 2008.

[14] S. E. F. De Avila, A. P. B. Lopes, A. da Luz, and A. de Albuquerque Araújo. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68, 2011.

[15] H. De Vries, F. Strub, J. Mary, H. Larochelle, O. Pietquin, and A. C. Courville. Modulating early visual processing by language. In *Advances in Neural Information Processing Systems*, pages 6594–6604, 2017.

[16] A. G. del Molino and M. Gygli. Phd-gifs: Personalized highlight detection for automatic gif creation. In *ACM International Conference on Multimedia*, pages 600–608, 2018.

[17] F. Dirfaux. Key frame selection to represent a video. In *IEEE International Conference on Image Processing*, pages 275–278, 2000.

[18] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[19] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *Advances in Neural Information Processing Systems*, pages 2121–2129, 2013.

[20] J. Gao, C. Sun, Z. Yang, and R. Nevatia. Tall: Temporal activity localization via language query. In *IEEE International Conference on Computer Vision*, pages 5277–5285, 2017.

[21] Y. Gao, T. Zhang, and J. Xiao. Thematic video thumbnail selection. In *IEEE International Conference on Image Processing*, pages 4333–4336, 2009.

[22] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin. Convolutional sequence to sequence learning. In *International Conference on Machine Learning*, pages 1243–1252, 2017.

[23] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2014.

[24] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[25] M. Gygli. Ridiculously fast shot boundary detection with fully convolutional neural networks. In *International Conference on Content-Based Multimedia Indexing*, pages 1–4, 2018.

[26] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In *European Conference on Computer Vision*, pages 505–520, 2014.

[27] M. Gygli, H. Grabner, and L. Van Gool. Video summarization by learning submodular mixtures of objectives. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3090–3098, 2015.

[28] M. Gygli, Y. Song, and L. Cao. Video2gif: Automatic generation of animated

gifs from video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1001–1009, 2016.

[29] S. Hasebe, M. Nagumo, S. Muramatsu, and H. Kikuchi. Video key frame selection by clustering wavelet coefficients. In *European Signal Processing Conference*, pages 2303–2306, 2004.

[30] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[31] X. Huang and S. Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *IEEE International Conference on Computer Vision*, pages 1510–1519, 2017.

[32] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5967–5976, 2017.

[33] A. Jaimes, T. Echigo, M. Teraguchi, and F. Satoh. Learning personalized video highlights from detailed mpeg-7 metadata. In *International Conference on Image Processing*, 2002.

[34] Y. Jiao, X. Yang, T. Zhang, S. Huang, and C. Xu. Video highlight detection via deep ranking modeling. In *Pacific-Rim Symposium on Image and Video Technology*, pages 28–39, 2017.

[35] H.-B. Kang. A scene-level analysis for video abstraction. In *Asian Conference on Computer Vision*, pages 81–86, 2002.

[36] H.-W. Kang and X.-Q. Chen. Space-time video montage. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1331–1338, 2006.

[37] H.-W. Kang and X.-S. Hua. To learn representativeness of video frames. In *ACM International Conference on Multimedia*, pages 423–426, 2005.

[38] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2698–2705, 2013.

[39] G. Kim and E. P. Xing. Reconstructing storyline graphs for image recommendation from web community photos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3882–3889, 2014.

[40] D. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.

[41] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

[42] R. Krishna, K. Hata, F. Ren, L. Fei-Fei, and J. C. Niebles. Dense-captioning events in videos. In *International Conference on Computer Vision*, pages 706–715, 2017.

[43] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager. Temporal convolutional networks for action segmentation and detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1003–1012, 2017.

[44] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1346–1353, 2012.

[45] X. Li, B. Zhao, and X. Lu. A general framework for edited video and raw video summarization. *IEEE Transactions on Image Processing*, 26(8):3652–3664, 2017.

[46] Y. Li and B. Merialdo. Multi-video summarization based on video-mmr. In *International Workshop on Image Analysis for Multimedia Interactive Services*, pages 1–4, 2010.

[47] Y. Li, L. Wang, T. Yang, and B. Gong. How local is the local diversity? reinforcing sequential determinantal point processes with dynamic ground sets for supervised video summarization. In *European Conference on Computer Vision*, pages 156–174, 2018.

[48] R. Lienhart, S. Pfeiffer, and W. Effelsberg. Video abstracting. *Communications of the ACM*, 40(12):54–62, 1997.

[49] C. Liu, Q. Huang, and S. Jiang. Query sensitive dynamic web video thumbnail generation. In *IEEE International Conference on Image Processing*, pages 2449–2452, 2011.

[50] M. Liu, X. Wang, L. Nie, X. He, B. Chen, and T.-S. Chua. Attentive moment retrieval in videos. In *ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 15–24, 2018.

[51] M.-Y. Liu, X. Huang, A. Mallya, T. Karras, T. Aila, J. Lehtinen, and J. Kautz. Few-shot unsupervised image-to-image translation. In *IEEE International Conference on Computer Vision*, pages 10550–10559, 2019.

[52] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. In *European Conference on Computer Vision*, pages 403–417, 2002.

[53] W. Liu, T. Mei, Y. Zhang, C. Che, and J. Luo. Multi-task deep visual-semantic embedding for video thumbnail selection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3707–3715, 2015.

[54] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[55] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2714–2721, 2013.

[56] J. Luo, C. Papin, and K. Costello. Towards extracting semantically meaningful key frames from personal video clips: from humans to computers. *IEEE Transactions on Circuits and Systems for Video Technology*, 19(2):289–301, 2008.

[57] Y.-F. Ma, L. Lu, H.-J. Zhang, and M. Li. A user attention model for video summarization. In *ACM International Conference on Multimedia*, pages 533–542, 2002.

[58] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial LSTM networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2982–2991, 2017.

[59] T. Mei, B. Yang, S.-Q. Yang, and X.-S. Hua. Video collage: presenting a video sequence using a single image. *The Visual Computer*, 25(1):39–51, 2009.

[60] P. Mundur, Y. Rao, and Y. Yesha. Keyframe-based video summarization using delaunay clustering. *International Journal on Digital Libraries*, 6(2):219–232, 2006.

[61] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang. Automatic video summarization by graph modeling. In *IEEE International Conference on Computer Vision*, pages 104–109, 2003.

[62] Y. Pan, T. Yao, T. Mei, H. Li, C.-W. Ngo, and Y. Rui. Click-through-based cross-view learning for image search. In *ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 717–726, 2014.

[63] R. Panda, A. Das, Z. Wu, J. Ernst, and A. K. Roy-Chowdhury. Weakly supervised summarization of web videos. In *IEEE International Conference on Computer Vision*, pages 3677–3686, 2017.

[64] R. Panda and A. K. Roy-Chowdhury. Collaborative summarization of topic-related videos. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4274–4283, 2017.

[65] A. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention

model for natural language inference. In *Conference on Empirical Methods in Natural Language Processing*, pages 2249–2255, 2016.

[66] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2332–2341, 2019.

[67] J. Pennington, R. Socher, and C. D. Manning. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[68] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *European Conference on Computer Vision*, pages 540–555, 2014.

[69] M. Rochan, M. K. K. Reddy, L. Ye, and Y. Wang. Adaptive video highlight detection by learning from user history. In *European Conference on Computer Vision*, 2020.

[70] M. Rochan and Y. Wang. Video summarization by learning from unpaired data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7894–7903, 2019.

[71] M. Rochan, L. Ye, and Y. Wang. Video summarization using fully convolutional sequence networks. In *European Conference on Computer Vision*, pages 358–374, 2018.

[72] M. Schuster and P. K. Kuldip. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997.

[73] A. Sharghi, A. Borji, C. Li, T. Yang, and B. Gong. Improving sequential determinantal point processes for supervised video summarization. In *European Conference on Computer Vision*, pages 533–550, 2018.

[74] A. Sharghi, B. Gong, and M. Shah. Query-focused extractive video summarization. In *European Conference on Computer Vision*, pages 3–19, 2016.

[75] A. Sharghi, J. S. Laurel, and B. Gong. Query-focused video summarization: Dataset, evaluation, and a memory network based approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2127–2136, 2017.

[76] A. F. Smeaton, P. Over, and W. Kraaij. Evaluation campaigns and trecvid. In *ACM International Workshop on Multimedia Information Retrieval*, pages 321–330, 2006.

[77] M. Soleymani. The quest for visual interest. In *ACM International Conference on Multimedia*, pages 919–922, 2015.

[78] Y. Song, M. Redi, J. Vallmitjana, and A. Jaimes. To click or not to click: Automatic selection of beautiful thumbnails from videos. In *ACM International on Conference on Information and Knowledge Management*, pages 659–668, 2016.

[79] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: Summarizing web

videos using titles. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 5179–5187, 2015.

[80] M. Sun, A. Farhadi, and S. Seitz. Ranking domain-specific highlights by analyzing edited videos. In *European Conference on Computer Vision*, pages 787–802, 2014.

[81] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[82] Y. Takahashi, N. Nitta, and N. Babaguchi. User and device adaptation for sports video content. In *IEEE International Conference on Multimedia and Expo*, pages 1051–1054, 2007.

[83] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3d convolutional networks. In *IEEE International Conference on Computer Vision*, pages 4489–4497, 2015.

[84] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 3(1):3, 2007.

[85] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.

[86] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis.

In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4105–4113, 2017.

[87] A. B. Vasudevan, M. Gygli, A. Volokitin, and L. Van Gool. Query-adaptive video summarization via quality-aware relevance estimation. In *ACM International Conference on Multimedia*, pages 582–590, 2017.

[88] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[89] O. Vinyals, M. Fortunato, and N. Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

[90] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[91] Z. Wang, M. Kumar, J. Luo, and B. Li. Extracting key frames from consumer videos using bi-layer group sparsity. In *ACM International Conference on Multimedia*, pages 1505–1508, 2011.

[92] Z. Wei, B. Wang, M. H. Nguyen, J. Zhang, Z. Lin, X. Shen, R. Mech, and D. Samaras. Sequence-to-segment networks for segment detection. In *Advances in Neural Information Processing Systems*, pages 3507–3516, 2018.

[93] W. Wolf. Key frame selection by motion analysis. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 1228–1231, 1996.

[94] C.-Y. Wu, M. Zaheer, H. Hu, R. Manmatha, A. J. Smola, and P. Krähenbühl. Compressed video action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 6026–6035, 2018.

[95] B. Xiong, Y. Kalantidis, D. Ghadiyaram, and K. Grauman. Less is more: Learning highlight detection from video duration. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1258–1267, 2019.

[96] T. Yao, T. Mei, and Y. Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 982–990, 2016.

[97] 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 *AAAI Conference on Artificial Intelligence*, 2018.

[98] Y. Yuan, L. Ma, J. Wang, W. Liu, and W. Zhu. Semantic conditioned dynamic modulation for temporal sentence grounding in videos. In *Advances in Neural Information Processing Systems*, pages 536–546, 2019.

[99] Y. Yuan, L. Ma, and W. Zhu. Sentence specified dynamic video thumbnail generation. In *ACM International Conference on Multimedia*, pages 2332–2340, 2019.

[100] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena. Self-attention generative adversarial networks. In *International Conference on Machine Learning*, pages 7354–7363, 2019.

[101] H. J. Zhang, J. Wu, D. Zhong, and S. W. Smoliar. An integrated system for content-based video retrieval and browsing. *Pattern recognition*, 30(4):643–658, 1997.

[102] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Summary transfer: Exemplar-based subset selection for video summarization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1059–1067, 2016.

[103] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In *European Conference on Computer Vision*, pages 766–782, 2016.

[104] K. Zhang, K. Grauman, and F. Sha. Retrospective encoders for video summarization. In *European Conference on Computer Vision*, pages 391–408, 2018.

[105] Y. Zhang, M. Kampffmeyer, X. Liang, M. Tan, and E. P. Xing. Query-conditioned three-player adversarial network for video summarization. In *British Machine Vision Conference*, 2018.

[106] Y. Zhang, M. Kampffmeyer, X. Liang, D. Zhang, M. Tan, and E. P. Xing. Dilated temporal relational adversarial network for generic video summarization. *Multimedia Tools and Applications*, 78(24):35237–35261, 2019.

[107] B. Zhao, X. Li, and X. Lu. Hierarchical recurrent neural network for video summarization. In *ACM International Conference on Multimedia*, pages 863–871, 2017.

[108] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos.

In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2513–2520, 2014.

[109] J. Zhao, M. Mathieu, and Y. LeCun. Energy-based generative adversarial network. In *International Conference on Learning Representations*, 2017.

[110] K. Zhou, Y. Qiao, and T. Xiang. Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward. In *AAAI Conference on Artificial Intelligence*, 2018.

[111] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *IEEE International Conference on Computer Vision*, pages 2242–2251, 2017.