Video Summarization Using Fully Convolutional Sequence Networks

Mrigank Rochan(✉), Linwei Ye, and Yang Wang

University of Manitoba, Winnipeg R3T 2N2, Canada
{mrochan,yel3,ywang}@cs.umanitoba.ca

Abstract. This paper addresses the problem of video summarization. Given an input video, the goal is to select a subset of the frames to create a summary video that optimally captures the important information of the input video. With the large amount of videos available online, video summarization provides a useful tool that assists video search, retrieval, browsing, etc. In this paper, we formulate video summarization as a sequence labeling problem. Unlike existing approaches that use recurrent models, we propose fully convolutional sequence models to solve video summarization. We firstly establish a novel connection between semantic segmentation and video summarization, and then adapt popular semantic segmentation networks for video summarization. Extensive experiments and analysis on two benchmark datasets demonstrate the effectiveness of our models.

Keywords: Video summarization
Fully convolutional neural networks · Sequence labeling

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. Video has become one of the most important form of visual data. Due to the sheer amount of video data, it is unrealistic for humans to watch these videos and identify useful information. 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! It is therefore becoming increasingly important to develop computer vision techniques that can enable efficient browsing of the enormous video data. In particular, video summarization has emerged as a promising tool to help cope with the overwhelming amount of video data.

Given an input video, the goal of video summarization is to create a shorter video that captures the important information of the input video. Video summarization can 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

© Springer Nature Switzerland AG 2018
V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11216, pp. 358–374, 2018.
https://doi.org/10.1007/978-3-030-01258-8_22

summary video that captures the important information from a long video, it will greatly reduce human efforts required in video surveillance. Video summarization can also provide better user experience in video search, retrieval, and understanding. Since short videos are easier to store and transfer, they can be useful for mobile applications. The summary videos can also help in many downstream video analysis tasks. For example, it is faster to run any other analysis algorithms (e.g. action recognition) on short videos.

In this paper, 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 [24,40] consider video summarization as a sequence labeling problem and solve the problem using a variant of recurrent neural networks known as the long short-term memory (LSTM) [11]. 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) [31] 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 [3,7,17] in action detection, audio synthesis, and machine translation showing that convolutional models can outperform recurrent models and can take full advantage of GPU parallelization.

In this paper, we propose to use fully convolutional networks for video summarization. Fully convolutional networks (FCN) [22] 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 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 paper, we develop our video summarization method based on popular semantic segmentation models such as FCN [22]. We call our approach the *Fully Convolutional Sequence Network (FCSN).*

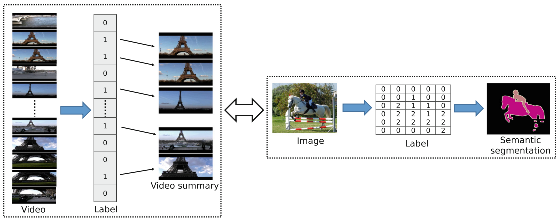

Fig. 1. An illustration of the relationship between video summarization and semantic segmentation. (*Left*) In video summarization, our goal is to select frames from an input video to generate the summary video. This is equivalent to assigning a binary label (0 or 1) to each frame in the video to indicate 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.

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.

The contributions of this paper are manifold. (1) To the best of our knowledge, we are the first to propose fully convolutional models for video summarization. (2) We establish a novel connection between two seemingly unrelated problems, namely video summarization and semantic segmentation. We then present a way to adapt popular semantic segmentation networks for video summarization. (3) We propose both supervised and unsupervised fully convolutional models. (4) Through extensive experiments on two benchmark datasets, we show that our model achieves state-of-the-art performance.

2 Related Work

Given an input video, video summarization aims to produce a shortened version that captures the important information in the video. There are various representations proposed for this problem including video synopsis [30], time-lapses [12,16,28], montages [13,35] and storyboards [8–10,18,21,24,38–40]. Our work is most related to storyboards which select a few representative video frames to summarize key events present in the entire video. Storyboard-based summarization has two types of outputs: keyframes [8,18,21] in which certain isolated frames are chosen to form the summary video, and keyshots [9,10,24,39,40] in which a set of correlated consecutive frames within a temporal slot are considered for summary generation.

Early work in video summarization mainly relies on hand-crafted heuristics. Most of these approaches are unsupervised. They define various heuristics to represent the importance or representativeness [14,15,18,23,26,27,34] of the frames and use the importance scores to select representative frames to form the summary video. Recent work has explored supervised learning approaches for video summarization [8–10,39,40]. These approaches use training data consisting of videos and their ground-truth summaries generated by humans. These supervised learning approaches tend to outperform early work on unsupervised methods, since they can implicitly learn high-level semantic knowledge that is used by humans to generate summaries.

Recently deep learning methods [24,32,40] are gaining popularity for video summarization. The most relevant works to ours are the methods that use recurrent models such as LSTMs [11]. The intuition of using LSTM is to effectively capture long-range dependencies among video frames which are crucial for meaningful summary generation. Zhang et al. [40] consider the video summarization task as a structured prediction problem on sequential data and model the variable-range dependency using two LSTMs. One LSTM is used for video sequences in the forward direction and the other for the backward direction. They further improve the diversity in the subset selection by incorporating a determinantal point process model [8,39]. Mahasseni et al. [24] propose an unsupervised generative adversarial framework consisting of the summarizer and discriminator. The summarizer is a variational autoencoder LSTM which first selects video frames and then decodes the output for reconstruction. The discriminator is another LSTM network that learns to distinguish between the input video and its reconstruction. They also extend their method to supervised learning by introducing a keyframe regularization. Different from these LSTM-based approaches, we propose fully convolutional sequence models for video summarization. Our work is the first to use fully convolutional models for this problem.

3 Our Approach

In this section, we first describe the problem formulation (Sect. 3.1). We then introduce our fully convolutional sequence model and the learning algorithm (Sect. 3.2). Finally, we present an extension of the basic model for unsupervised learning of video summarization (Sect. 3.3).

3.1 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 [5,8,25,40] or keyshots [9,10,29,34,40]. 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 [9,34] 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 paper, 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. 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.

3.2 Fully Convolutional Sequence Networks

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 [40] 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 FCSN for video summarization. In this section, we describe a FCSN based on a popular semantic segmentation network, namely FCN [22]. 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 FCSN for video summarization.

SUM-FCN: FCN [22] 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.

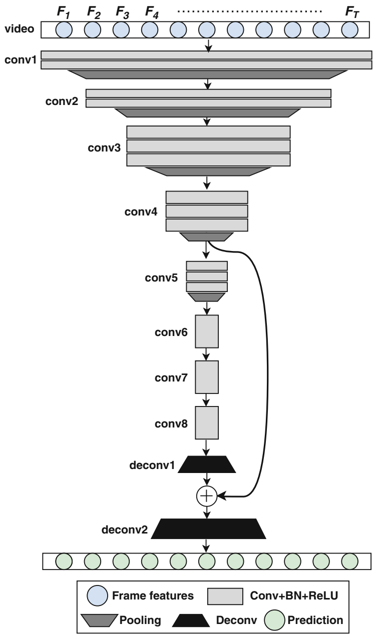

Fig. 2. The architecture of SUM-FCN. It is based on the popular semantic segmentation architecture FCN [22]. Unlike FCN, SUM-FCN performs convolution, pooling and deconvolution operation across time.

Figure 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 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 [22]. 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 [6].

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:

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

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.

3.3 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 [24, 40]. 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 [42] $\mathcal{L}_{div}$ to enforce diversity among selected keyframes. We define $\mathcal{L}_{div}$ as the mean of the pairwise similarity between the selected Y keyframes:

$$\mathcal{L}_{div} = \frac{1}{|Y|(|Y|-1)} \sum_{t \in Y} \sum_{t' \in Y, t' \neq t} d(f_t, f_{t'}), \text{ where } (f_t, f_{t'}) = \frac{f_t^T f_{t'}}{\|f_t\|_2 \|f_{t'}\|_2} \tag{2}$$

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.

4 Experiments

In this section, we first introduce the datasets in Sect. 4.1. We then discuss the implementation details and setup in Sect. 4.2. Lastly, we present the main results in Sect. 4.3 and additional ablation analysis in Sect. 4.4.

4.1 Datasets

We evaluate our method on two benchmark datasets: SumMe [9] and TVSum [34]. The SumMe dataset is a collection of 25 videos that cover a variety of events (e.g. sports, holidays, etc.). The videos in SumMe are 1.5 to 6.5 min in length. The TVSum dataset contains 50 YouTube videos of 10 different categories (e.g. making sandwich, dog show, changing vehicle tire, etc.) from the TRECVid Multimedia Event Detection (MED) task [33]. The videos in this dataset are typically 1 to 5 min in length.

Since training a deep neural network with small annotated datasets is difficult, previous work [40] has proposed to use additional videos to augment the datasets. Following [40], we use 39 videos from the YouTube dataset [5] and 50 videos from the Open Video Project (OVP) dataset [2,5] 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 Sect. 4.2 on how we handle different formats of ground-truth annotations.

4.2 Implementation Details and Setup

Features: Following [40], we uniformly downsample the videos to 2 fps. Next, we take the output of the *pool*5 layer in the pretrained GoogleNet [36] 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 [37]). We use GoogleNet features mainly because they are used in previous work [24,40] and will allow fair comparison in the experiments.

Ground-Truth: Since different datasets provide the ground-truth annotations in various format, we follow [8,40] 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 [9,10,40]) 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 [40] which includes the following steps: (1) temporally segment a video using KTS [29] 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 [34] 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 [40]. 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 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 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 Sect. 4.4).

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.

Table 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 [34,40]. †We follow [40] to convert multiple frame-level scores to keyshots. §Following [8,40], 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 [24,40]), so we do not test our methods on them

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§	-

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 Sect. 4.4, we experiment with sampling the videos to a longer length. We also experiment with directly operating on original non-sampled (variable length) videos in Sect. 4.4.

We follow [24,40] to convert predicted keyframes to keyshots so that we can perform fair comparison with other methods. We first apply KTS [29] 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 [34] to ensure that the produced keyshot-based summary is of maximum 15% in length of the original test video.

Evaluation Metrics: Following [24,40], 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:

$$P = \frac{|S_O \cap S_G|}{|S_O|}, R = \frac{|S_O \cap S_G|}{|S_G|} \tag{3}$$

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 [10,34,40] to calculate the metric for videos that have multiple ground-truth summaries.

Experiment Settings: Similar to previous work [39,40], 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 rest 80% for training and validation. Since the data is randomly splitted, 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% data for testing and leave the rest 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 with other datasets is well-known in computer vision. This is usually referred as data augmentation. Recent methods [24,40] 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. [39,40]. 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.

4.3 Main Results and Comparisons

We compare the performance of our approach (SUM-FCN) with prior methods on the SumMe dataset in Table 2. Our method outperforms other state-of-the-art approaches by a large margin.

Table 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. Note that Zhang et al. [40] (vsLSTM) use frame-level importance scores and Zhang et al. [40] (dppLSTM) use both keyframe-based annotation

Table 2. Comparison of summarization performance (F-score) between SUM-FCN and other approaches on the SumMe dataset under different settings

Dataset	Method	Standard supervised	Augmented	Transfer
SumMe	Gygli et al. [9]	39.4	–	–
	Gygli et al. [10]	39.7	–	–
	Zhang et al. [39]	40.9	41.3	38.5
	Zhang et al. [40] (vsLSTM)	37.6	41.6	40.7
	Zhang et al. [40] (dppLSTM)	38.6	42.9	41.8
	Mahasseni et al. [24] (supervised)	41.7	43.6	–
	Li et al. [19]	43.1	–	–
	SUM-FCN (ours)	**47.5**	**51.1**	**44.1**

and frame-level importance scores. But we only use keyframe-based annotation in our method. Previous method [40] 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.

Table 3. Performance (F-score) of SUM-FCN and other approaches on the TVSum dataset. †Zhang et al. [40] (vsLSTM) use frame-level importance scores. ‡Zhang et al. [40] (dppLSTM) use both frame-level importance scores and keyframes in their method. Different from these two methods, our method only uses keyframe-based annotations

Dataset	Method	Standard supervised	Augmented	Transfer
TVSum	Zhang et al. [40] (vsLSTM)	54.2	57.9	56.9†
	Zhang et al. [40] (dppLSTM)	54.7	59.6	**58.7**‡
	Mahasseni et al. [24] (supervised)	56.3	**61.2**	–
	Li et al. [19]	52.7	–	–
	SUM-FCN (ours)	**56.8**	59.2	58.2

4.4 Analysis

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

Unsupervised SUM-FCN$_{unsup}$: Table 4 compares the performance of SUM-FCN$_{unsup}$ with the other unsupervised methods in the literature. SUM-FCN$_{unsup}$ achieves the 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.

SUM-DeepLab: To demonstrate the generality of FCSN, we also adapt DeepLab [4] (in particular, DeepLabv2 (VGG16) model), another popular

Table 4. Performance (F-score) comparison of SUM-FCN$_{unsup}$ with state-of-the-art unsupervised methods

Dataset	[5]	[20]	[14]	[34]	[41]	[24]	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**

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 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 5).

Table 5. Performance (F-score) of SUM-DeepLab in different settings. We include the performance of SUM-FCN (taken from Tables 2 and 3) in brackets. We also replace the bilinear upsampling with learnable deconvolutional layer and report the result in the transfer setting (last column)

Dataset	Standard 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**

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 6 (first two columns) shows the results of our models for this case.

Compared with $T = 320$ (shown in brackets in Table 6), the performance with $T = 640$ is similar. This shows that the video length is not an issue for our proposed fully convolutional models.

Table 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 2, 3 and 5)

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

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 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, we show example video summaries (good and poor) produced by SUM-FCN on two videos in the SumMe [9] dataset.

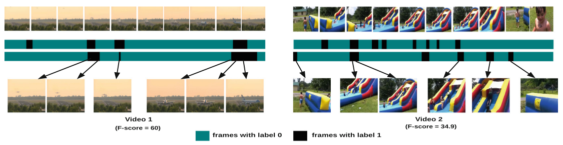

Fig. 3. Example summaries for two videos in the SumMe [9] dataset. The black bars on the green background show the frames selected to form the summary video. For each video, we show the ground-truth (*top bar*) and the predicted labels (*bottom bar*). (Color figure online)

5 Conclusion

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 segmentation 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. As future work, we plan to explore more recent semantic segmentation models and develop their counterpart models in video summarization.

Acknowledgments. This work was supported by NSERC, a University of Manitoba Graduate Fellowship, and the University of Manitoba GETS program. We thank NVIDIA for donating some of the GPUs used in this work.

References

1. Cisco visual networking index: Forecast and methodology, 2016–2021. https://www.cisco.com/
2. Open video project. https://open-video.org/
3. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271 (2018)
4. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**, 834–848 (2017)
5. De Avila, S.E.F., Lopes, A.P.B., da Luz, A., de Albuquerque Araújo, A.: VSUMM: a mechanism designed to produce static video summaries and a novel evaluation method. Pattern Recognit. Lett. **32**(1), 56–68 (2011)
6. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: IEEE International Conference on Computer Vision (2015)
7. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: International Conference on Machine Learning (2017)
8. Gong, B., Chao, W.L., Grauman, K., Sha, F.: Diverse sequential subset selection for supervised video summarization. In: Advances in Neural Information Processing Systems (2014)
9. Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 505–520. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_33
10. Gygli, M., Grabner, H., Van Gool, L.: Video summarization by learning submodular mixtures of objectives. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
11. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
12. Joshi, N., Kienzle, W., Toelle, M., Uyttendaele, M., Cohen, M.F.: Real-time hyperlapse creation via optimal frame selection. ACM Trans. Graph. **34**(4), 63 (2015)
13. Kang, H.W., Chen, X.Q.: Space-time video montage. In: IEEE Conference on Computer Vision and Pattern Recognition (2006)

14. Khosla, A., Hamid, R., Lin, C.J., Sundaresan, N.: Large-scale video summarization using web-image priors. In: CVPR (2013)
15. Kim, G., Xing, E.P.: Reconstructing storyline graphs for image recommendation from web community photos. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
16. Kopf, J., Cohen, M.F., Szeliski, R.: First-person hyper-lapse videos. ACM Trans. Graph. **33**(4), 78 (2014)
17. Lea, C., Flynn, M.D., Vidal, R., Reiter, A., Hager, G.D.: Temporal convolutional networks for action segmentation and detection. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
18. Lee, Y.J., Ghosh, J., Grauman, K.: Discovering important people and objects for egocentric video summarization. In: IEEE Conference on Computer Vision and Pattern Recognition (2012)
19. Li, X., Zhao, B., Lu, X.: A general framework for edited video and raw video summarization. IEEE Trans. Image Process. **26**(8), 3652–3664 (2017)
20. Li, Y., Merialdo, B.: Multi-video summarization based on video-MMR. In: Workshop on Image Analysis for Multimedia Interactive Services (2010)
21. Liu, D., Hua, G., Chen, T.: A hierarchical visual model for video object summarization. IEEE Trans. Pattern Anal. Mach. Intell. **32**(12), 2178–2190 (2010)
22. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
23. Lu, Z., Grauman, K.: Story-driven summarization for egocentric video. In: IEEE Conference on Computer Vision and Pattern Recognition (2013)
24. Mahasseni, B., Lam, M., Todorovic, S.: Unsupervised video summarization with adversarial LSTM networks. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
25. Mundur, P., Rao, Y., Yesha, Y.: Keyframe-based video summarization using delaunay clustering. Int. J. Digit. Libr. **6**(2), 219–232 (2006)
26. Ngo, C.W., Ma, Y.F., Zhang, H.J.: Automatic video summarization by graph modeling. In: IEEE International Conference on Computer Vision (2003)
27. Panda, R., Roy-Chowdhury, A.K.: Collaborative summarization of topic-related videos. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
28. Poleg, Y., Halperin, T., Arora, C., Peleg, S.: EgoSampling: fast-forward and stereo for egocentric videos. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
29. Potapov, D., Douze, M., Harchaoui, Z., Schmid, C.: Category-specific video summarization. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8694, pp. 540–555. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10599-4_35
30. Pritch, Y., Rav-Acha, A., Peleg, S.: Nonchronological video synopsis and indexing. IEEE Trans. Pattern Anal. Mach. Intell. **30**(11), 1971–1984 (2008)
31. Schuster, M., Kuldip, P.K.: Bidirectional recurrent neural networks. IEEE Trans. Signal Process. **45**, 2673–2681 (1997)
32. Sharghi, A., Laurel, J.S., Gong, B.: Query-focused video summarization: dataset, evaluation, and a memory network based approach. In: IEEE Conference on Computer Vision and Pattern Recognition (2017)
33. Smeaton, A.F., Over, P., Kraaij, W.: Evaluation campaigns and TRECVid. In: Multimedia Information Retrieval. ACM (2006)

34. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: TVSum: summarizing web videos using titles. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
35. Sun, M., Farhadi, A., Taskar, B., Seitz, S.: Salient montages from unconstrained videos. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8695, pp. 472–488. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10584-0_31
36. Szegedy, C., et al.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (2015)
37. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: IEEE International Conference on Computer Vision (2015)
38. Yang, H., Wang, B., Lin, S., Wipf, D., Guo, M., Guo, B.: Unsupervised extraction of video highlights via robust recurrent auto-encoders. In: IEEE International Conference on Computer Vision (2015)
39. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Summary transfer: examplar-based subset selection for video summarization. In: IEEE Conference on Computer Vision and Pattern Recognition (2016)
40. Zhang, K., Chao, W.-L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 766–782. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_47
41. Zhao, B., Xing, E.P.: Quasi real-time summarization for consumer videos. In: IEEE Conference on Computer Vision and Pattern Recognition (2014)
42. Zhao, J., Mathieu, M., LeCun, Y.: Energy-based generative adversarial network. In: International Conference on Learning Representations (2017)