Unsupervised Extraction of Video Highlights Via Robust Recurrent Auto-encoders

Huan Yang[1]* Baoyuan Wang[2] Stephen Lin[3] David Wipf[3] Minyi Guo[1]
Baining Guo[3]
[1]Shanghai Jiao Tong University [2]Microsoft Technology & Research [3]Microsoft Research

Abstract

With the growing popularity of short-form video sharing platforms such as Instagram and *Vine, there has been an increasing need for techniques that automatically extract highlights from video. Whereas prior works have approached this problem with heuristic rules or supervised learning, we present an unsupervised learning approach that takes advantage of the abundance of user-edited videos on social media websites such as YouTube. Based on the idea that the most significant sub-events within a video class are commonly present among edited videos while less interesting ones appear less frequently, we identify the significant sub-events via a robust recurrent auto-encoder trained on a collection of user-edited videos queried for each particular class of interest. The auto-encoder is trained using a proposed shrinking exponential loss function that makes it robust to noise in the web-crawled training data, and is configured with bidirectional long short term memory (LSTM) [5] cells to better model the temporal structure of highlight segments. Different from supervised techniques, our method can infer highlights using only a set of downloaded edited videos, without also needing their pre-edited counterparts which are rarely available online. Extensive experiments indicate the promise of our proposed solution in this challenging unsupervised setting.*

1. Introduction

Short-form video has become a popular way for users to share their experiences on social media platforms such as *YouTube* and *Facebook*. With a well-crafted video, the user's experience can be quickly conveyed without testing the attention span of viewers. However, manually producing a highlight clip from a lengthy video, such as those captured with wearable devices like *GoPro* cameras, can be a time-consuming and laborious task, especially on small form-factor display devices such as smart phones. An automated tool for generating highlight clips is thus immensely desirable such that the user need only to deal with content capture.

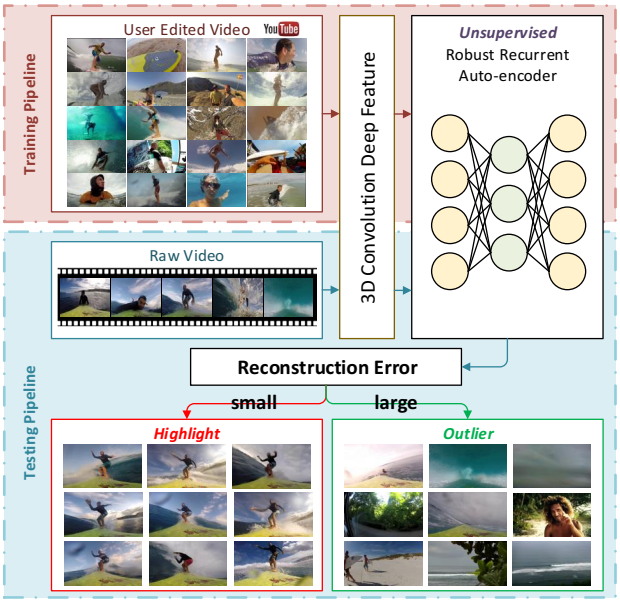

Figure 1. Overall system pipeline

Previous techniques address this problem either by limiting the scope to a particular context or through the use of supervised learning. The first type of method generally employs heuristic rules designed for a certain type of video, such as for broadcast sports [20, 18, 25, 31]. Though effective for their targeted settings, these techniques may not generalize well to generic, unstructured videos. In contrast, methods based on supervised learning rely on pairs of edited and raw source videos [24] to infer highlight actions. Collecting such video pairs, however, can be a challenge. Although there exists a considerable amount of video data on the web, users typically do not upload both the raw and edited versions of a video.

In this work, we propose an *unsupervised* approach for

*This work was done when Huan Yang was an intern at Microsoft Research

1550-5499/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCV.2015.526

generating highlight clips, using only edited videos. On the web, there are seemingly countless short-form videos that have been edited by thousands of users to contain mainly highlight sub-events. Our method capitalizes on this wealth of data by web crawling for videos in a given domain (e.g., "surfing"), and modeling the highlights from them by inferring their common features. In this way, their raw counterparts are not needed, making it easy to scale up and collect more training data. Additionally, since the videos have been edited by a large community of users, the risk of building a biased highlight model is greatly reduced in comparison to using a training set constructed from a small number of users.

There exist significant challenges with this approach: (1) Although most people have a common notion of what the highlights should be in a certain video domain such as "surfing", there nevertheless may exist subjective differences among users (e.g., whether entering the water is highlight-worthy). (2) A query on a given keyword, such as "GoPro surfing", may return some noisy results that are not relevant to the targeted domain. (3) No information other than the queried videos themselves are available to be leveraged. Unlike in previous supervised learning approaches [24, 19, 4, 2], there are no unedited counterpart videos that can be used to identify what is and is not important to keep in a highlight clip.

To address these issues, we propose to identify and model highlights as the most common sub-events among the queried videos and remove uninteresting or idiosyncratic snippet selections that occur relatively infrequently. Our method accomplishes this via an auto-encoding recursive neural network (RNN) [28] that is trained from positive examples to reconstruct highlight input instances accurately while non-highlights are not. Intuitively, since highlights are assumed to occur much more frequently among the queried videos, they will have clustered distributions in the feature space while non-highlights occur as outliers.

We formulate the auto-encoder with two main features. Since training data crawled from the web is generally noisy (containing some amount of negative examples), we propose a novel shrinking exponential loss function that makes the auto-encoder training robust to noisy data. With the shrinking exponential loss, outliers are gradually identified in the training data and their influence in the auto-encoder training is progressively reduced. The other main feature accounts for the temporal structure of video highlights (e.g., standing up on the surfboard, riding the wave, and then falling into the ocean). To take advantage of this contextual dependency, we construct the auto-encoder with bidirectional long short term memory (LSTM) [5] cells, which have been shown in areas such as speech recognition [3] to effectively model long-range context in time-series data.

The main technical contributions of this work are the formulation of video highlight detection as an unsupervised learning problem that takes advantage of the abundance of short-form video on the web, and modeling video highlight structure through a robust recurrent auto-encoder with a shrinking exponential loss function and bidirectional LSTM cells. With the proposed unsupervised technique, we show promising results that approach the quality of supervised learning but without the burden of collecting pre- and post-edit video pairs.

2. Related Work

As defined in [24], a video highlight is a moment of major or special interest in a video. Generating highlight clips is thus different from the task of video summarization, which instead accounts for factors such as "diversity" and "representativeness" to convey a brief but comprehensive synopsis of a video. Despite this different goal, we review methods for video summarization in addition to video highlight detection because of similarities between the two topics.

Video Highlight Detection Traditionally, video highlight detection has primarily been focused on broadcast sports videos [20, 18, 25, 31]. These techniques usually employ features that are specific to a given sport and the structure of sports broadcasts, and are therefore hard to generalize to the more generic videos of ordinary users. Recently, the scope of highlight detection was expanded to a broad range of videos in [24], where a latent SVM model was proposed to rank highlight segments ahead of less interesting parts through supervised learning. Good results have been demonstrated with this approach on action videos such as those captured by GoPro cameras. However, the supervised learning requires each training example to be a video pair composed of an edited video and its corresponding raw source video. Such training pairs are difficult to collect in large quantities, since users tend to upload/share only the edited versions, as the source videos generally are too long for general consumption. This makes large scale training mostly impractical for this approach. In addition, it employs a computationally-intensive feature representation, namely dense trajectories [29], which involves computing dense optical flows and extracting low-level features such as HoG, HoF and MBH prior to Fisher vector encoding. By contrast, our method performs unsupervised learning from edited videos only, and utilizes generic deep learning features which are more computationally efficient and more accurate in characterizing both appearance and motion.

Video Summarization A comprehensive review of video summarization can be found in [27]. Among recent methods, several are guided by saliency-based properties such

as attention [13], interestingness [17, 7, 4], and important people and objects [9]. However, the most salient frames in a video do not necessarily correspond to its highlights, which depend heavily on the video domain. Others aim to provide a comprehensive synopsis based on connectivity of sub-events [12] or diversity of video segments [32, 10]. While this helps to provide a complete overview of a video, a highlight clip instead is focused on only certain segments that are specific to the video domain, while discarding the rest.

Video summarization techniques have also employed supervised learning. Methods along this direction use category-specific classifiers for importance scoring [19] or learn how to select informative and diverse video subsets from human-created summaries [2]. These supervised techniques have led to state-of-the-art video summarization results, but are not suitable for highlight clip generation. In addition, it is generally feasible to have only a limited number of users to annotate training videos, which may lead to a biased summarization model. Our method instead learns from videos pooled from many users on the web to obtain a highlight model less influenced by the particular preferences of certain people.

Novelty Detection The one-class learning of our recurrent auto-encoder is related to works on novelty detection, which aim to identify outliers from an observed class. In [15], novelty detection is performed for audio features using an auto-encoder with LSTM. Our concurrent work deals instead with RGB video data, for which meaningful features are more challenging to extract. We address this through temporal video segmentation, extraction of high-level spatial-temporal features for each segment, and then temporal pooling, before feeding to the auto-encoder. Moreover, we introduce a novel shrinking loss function to address the noisy training data in our context.

There exist other unsupervised novelty detection techniques that could potentially be applied to our problem, such as the unsupervised one-class learning in [11] or outlier-robust PCA [30]. In our work, we chose the auto-encoder as our basic unsupervised framework because its properties are well-suited to our application, such as scalability, easy parallelization, and seamless integration with LSTM cells. How to customize other novelty detection techniques for video highlight detection is a potential direction for future investigation.

3. Auto-Encoder-Based Removal of Outliers

An auto-encoder is an artificial neural network [21] that is trained to reconstruct its own input. A common use of auto-encoders is for dimensionality reduction, where if the hidden layers have fewer nodes than the input/output layers, then the activations of the final hidden layer can be taken as a compressed representation of the original signal. An auto-encoder operates by taking an input vector $x \in [0,1]^d$ and first mapping it to a hidden layer $h \in [0,1]^{d'}$ through a deterministic function $f_\theta(x) = s(Wx + b)$, parameterized by $\theta = \{W, b\}$ where W is a $d' \times d$ weight matrix, b is a bias vector, and s is the activation function, such as a sigmoid or rectified linear unit (ReLU). This hidden layer is then mapped to an output layer $y \in [0,1]^d$ with the same number of nodes as the input layer through another function $g_{\theta'}(x) = s(W'x + b')$, with s being a linear function at this layer. Thus, $y = g_{\theta'}(f_\theta(x))$. Backpropagation with stochastic gradient descent (SGD) is employed to optimize the parameters θ and θ' via the following loss function

$$(\theta^*, \theta'^*) = \arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(x^i, y^i) \quad (1)$$

$$= \arg\min_{\theta,\theta'} \frac{1}{n} \sum_{i=1}^{n} L(x^i, g_{\theta'}(f_\theta(x^i))), \quad (2)$$

where L is generally defined as the squared error $L(x,y) = \|x - y\|^2$ and each x^i is a training sample. When $d' < d$, the auto-encoder acts as a compression neural network that works surprisingly well for single-class document classification [14] and novelty detection [6]. The key idea is that inlier (or positive) instances are expected to be faithfully reconstructed by the auto-encoder, while outliers (or negative) instances are not. So one can classify an unseen instance by checking its reconstruction error from the auto-encoder. Our work is partially inspired by the applications of auto-encoders for novelty detection, and we present two significant modifications to tailor them for our highlight detection problem.

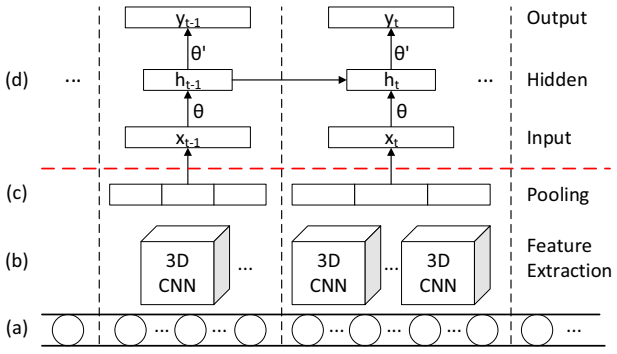

Figure 2. Conceptual illustration of our overall pipeline and architecture. (a) Each video is first segmented into multiple short snippets. (b) Then we apply a pre-trained 3D convolution neural network model [26] to extract spatial-temporal features. (c) This is followed by a temporal pooling scheme that respects the local ordering structure within each snippet. (d) The robust recurrent auto-encoder with the proposed companion loss is then employed to capture the long range contextual structure.

4. Our Approach

In this section, we introduce a new domain-specific video highlight system. Our core idea is to leverage the wealth of crowd-sourced video data from the web and automatically learn a parametric highlight detection model. Before describing the technical details, we first introduce the overall system pipeline, illustrated in Fig. 1.

4.1. Overview

Acquisition of Training Data: Our system starts with a simple video crawling step. Given the keyword for a specific domain, such as "GoPro Surfing", our system automatically retrieves a large number of videos from YouTube with this keyword. We restrict this search to only short-form videos (i.e., videos less than four minutes long), since such videos from social media are likely to have been edited by the user. With this approach, we can easily build a large-scale training set edited by many different people. Since our system learns what is a highlight based on commonalities among different videos, having a diverse user pool helps to avoid biases in this inference. Let us denote the training video set as $S = \{v_1, v_2, ..., v_N\}$. Our system then automatically models the highlights in S through the use of a proposed auto-encoder.

Temporal Segmentation: A highlight can be defined as a motion or moment of interest with respect to the video domain context. So we first segment each video $v_i, i \in [1, N]$ into multiple non-uniform snippets using an existing temporal segmentation algorithm [19]. We added a constraint to the segmentation algorithm to ensure that the number of frames within each snippet lies in the range of $[48, 96]$. The segmented snippets serve as the basic units for feature extraction and subsequent learning and inference (Fig. 2(a)). After segmentation, a highlight sequence in the edited video might correspond to one or multiple consecutive snippets. At runtime, our system outputs the highlight confidence score for each snippet within the input video.

Feature Representation: Recent work in deep learning has revealed that features extracted at higher layers of a convolutional neural network are generic features that have good transfer learning capabilities across different domains [1, 33, 8, 23, 26]. An advantage of using deep learning features is that there exist accurate, large-scale datasets such as Places [33] and One-million Sports [8] from which they can be extracted. In addition, GPU-based extraction of such features is much faster than that for the traditional bag-of-words and Fisher vector models. For example, C3D features [26] are $50\times$ faster to extract than dense trajectories [29]. We therefore extract C3D features, by taking sets of 16 input frames, applying 3D convolutional filters, and extracting the responses at layer "FC6" as suggested in [26] (Fig. 2(b)). This is followed by a temporal mean pooling scheme to maintain the local ordering structure within a snippet (Fig. 2(c)). Then the pooling result serves as the final input feature vector to be fed into the auto-encoder (Fig. 2(d)).

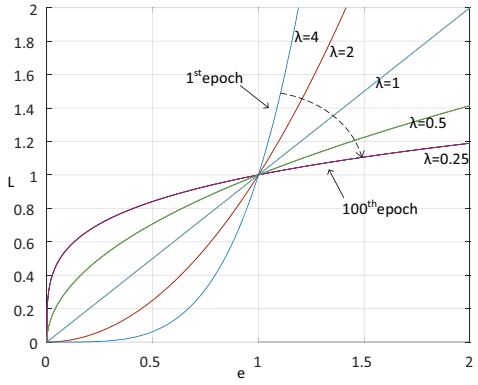

Figure 3. Unlike the squared loss used in standard auto-encoders, we propose a more general exponential loss $L = e^{\lambda}$ with its exponential parameter λ shrinking during the course of training. The horizontal axis e represents the reconstruction error, while the vertical axis L signifies the loss.

Unsupervised Learning: After building the representation for each snippet, we learn a discriminative model for highlight detection using a novel robust recurrent auto-encoder. It is assumed that the collected training set S for each domain contains coherence in the highlights so that they can be distinguished from the remaining parts by reconstruction error. We note that this cannot be treated as a multiple instance learning problem. Since a video does not necessarily contain at least one highlight snippet, such as when the video is actually unrelated to the keyword, the bag and instance relationship is hard to define.

4.2. Robust Auto-encoder Via Shrinking Exponential Loss

In training an auto-encoder, it is assumed that the training data consists only of positive instances, which the auto-encoder learns to replicate as output. However, since we obtain our training data through web crawling, we cannot guarantee that the data is free of negative instances. To train an auto-encoder that is robust to such noise, we propose a shrinking exponential loss function that helps to reduce the influence of negative examples. This function is defined by

$$L(x, y) = (\| x - y \|_2^2)^{\lambda} \quad (3)$$

$$\lambda = f(epo), \quad (4)$$

where λ is a function f of the current epoch number epo, and L is equivalent to the standard squared loss when $\lambda = 1$.

With backpropagation for network training, an example that has a large loss gradient will contribute more to the training than other examples whose corresponding loss gradient is small. Since network parameters are randomly initialized, all the examples will generally have a large loss at the beginning. So to expedite convergence in the early stages of training, we utilize a relatively large value of λ, which magnifies the loss gradients. As the positive examples share commonalities and are assumed to be more clustered relative to the negative examples, the network parameters will start to converge in a manner such that the positive examples become more accurately reconstructed. At the same time, it is desirable to shrink the exponent λ in order to decrease the influence of negative examples, which on average have larger loss gradients because of their more dispersive nature.

To accomplish this we define f to be monotonically decreasing with respect to epo as shown in Figure 3, with values greater than 1 in early stages to promote convergence, and shrinking to less than 1 in later stages to reduce the impact of outliers with higher reconstruction error. In our work, we empirically define f such that λ varies linearly in the range of $[e, s]$, with $e \in (0, 1]$ and $s \geq 1$, giving

$$f(epo) = s - \frac{epo * (s - e)}{\Gamma}, \tag{5}$$

where Γ is the total number of training epochs. As demonstrated later in our experiments, this formulation of our shrinking exponential loss provides greater robustness to negative examples than the standard squared loss for which λ is fixed to 1.

4.3. Recurrent Auto-Encoder with LSTM Cells

Each highlight snippet has a certain degree of dependence on its preceding and even subsequent frames. Taking surfing as an example, a surfer must first stand up on the board before riding a wave. The "stand up" action thus provides contextual information that can help to discriminate subsequent surfing highlights. In this work, we take advantage of such temporal dependencies through the use of long short term memory cells.

Given an input sequence $x = (x_1, x_2, ..., x_T), x_t \in R^d, t \in [1, T]$, a recurrent neural network (RNN) designed as an auto-encoder needs to first compute a hidden vector sequence $h = (h_1, h_2, ..., h_T), h_t \in R^{d'}, d' < d$ such that it outputs a reconstructed sequence $y = (y_1, y_2, ..., y_T)$ where $y_t \approx x_t$. This can be solved through iterations of the following equations:

$$h_t = \mathcal{H}(W_{ih}x_t + W_{hh}h_{t-1} + b_h) \tag{6}$$
$$y_t = W_{ho}h_t + b_o \tag{7}$$

where W_{ih} and W_{hh} denote the input-hidden and hidden-hidden weighting matrices, and b_h and b_o represent bias vectors. $\mathcal{H}$ is the hidden layer activation function, usually chosen as an element-wise sigmoid.

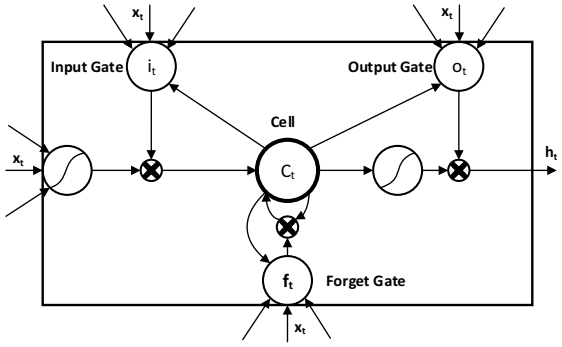

Figure 4. Long short term memory cell (regenerated from [3]).

For time-series data, it has been found that LSTM cells [5] are more effective at finding and modeling long-range context along a sequence, as shown in recent works on speech recognition [3] and human action recognition [16]. Figure 4 shows a typical structure of a LSTM cell, which operates by learning gate functions that determine whether an input is significant enough to remember, whether it should be forgotten, and when it should be sent to output. By storing information over different time ranges in this manner, a LSTM-RNN is better able to classify time-series data than a standard RNN. Inspired by these works, we propose to integrate LSTM cells as the hidden nodes in the auto-encoder network. With LSTM cells, $\mathcal{H}$ is then defined by the following composite functions:

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{8}$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{9}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{10}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{11}$$
$$h_t = o_t \tanh(c_t) \tag{12}$$

where σ is the logistic sigmoid function, and i, f, o are respectively the *input gate, forget gate and output gate*, which take scalar values between 0 and 1. c denotes the *cell* activation vectors which have the same size as the hidden vector h. The terms in the W matrices represent the connections, for example, with W_{xi} denoting the input-input gate matrix and W_{hf} representing the hidden-forget gate matrix.

In practice, we use bidirectional LSTM cells to model both forward and backward dependencies. For further details on LSTM, please refer to [5] and [3].

5. Experiments

5.1. Datasets

Edited Videos for Training Set As mentioned in Sec. 4.1, our system can automatically harvest domain-specific datasets from the web using keywords and other

	# of train videos	# of test videos	Coverage of train set	H-ratio of train set
freeride	912	27	0.63	0.33
parkour	781	29	0.83	0.32
skating	940	34	0.67	0.26
skiing	945	50	0.83	0.32
skydiving	762	30	0.83	0.38
surfing	928	52	0.80	0.31
swimming	1015	29	0.73	0.25

Table 1. Statistical information on our dataset

search conditions. Unlike previous data crawling systems such as in [24], ours need only obtain short-form videos edited by users, and not their corresponding raw versions. The editing operations may have been applied either by post-processing or through selective capture. For evaluating the performance of our approach, we have crawled more than 6500 short-form videos totaling about 13800 minutes from YouTube[1]. The search terms include *freeride, parkour, skating, skiing, skydiving, surfing and swimming*. Temporal segmentation of the videos yields 442075 snippets. Compared with the training set used in [24], ours is more than $10\times$ longer, and can easily be expanded further.

To obtain a better sense of how this set correlates with the underlying highlights, we present two quantitative measurements, *Coverage* and *H-ratio*. *Coverage* refers to the percentage of videos that contain at least one highlight snippet (the basic unit of segmentation), while *H-ratio* is the percentage of highlight snippets among all the snippets within the set. We calculate these measures from a randomly selected subset of 30 videos from each domain with manual highlight annotations. The statistics of our dataset in Table 1 show the *Coverage* value to be about 70% and the *H-ratio* about 30% in each domain, indicating that users indeed tend to share edited videos which have a significant amount of highlight content.

Raw Videos for Testing Set For each domain, we also manually collected about 30 raw videos (see third column of Table 1) which do not correspond to the training videos, and asked six people to annotate the highlights for each video. A snippet is considered to be a highlight only if they were labeled as such by at least four of the people. Annotations were collected by simply having the users mark each snippet as highlight or not. The total length of the testing videos is about 700 minutes, which is 2.5x longer than the testing data used by [24].

[1]We use the tool "youtube-dl" in http://rg3.github.io/youtube-dl/ to crawl for videos using the domain name and with a search condition of less than four minutes.

	Places CNN [4]	C3D [26]
freeride	0.241	0.302
parkour	0.323	0.425
skating	0.310	0.304
skiing	0.462	0.388
skydriving	0.337	0.433
surfing	0.501	0.539
swimming	0.350	0.320
Overall mAP	0.361	0.387

Table 2. Comparison of mean average precision (mAP) between 2D and 3D CNN features with a standard auto-encoder. Both types of features are 4096-D vectors. The Places CNN features were temporally pooled by dividing each snippet into two uniform subsnippets and performing mean pooling on each, while the C3D features were simply mean pooled within each whole snippet.

5.2. Implementation & Evaluation Details

Our system was implemented in C++ and runs on a workstation with a 3.1GHz dual-core CPU and an Nvidia Tesla K40 graphics card. For all the training and testing videos, we first segment them into multiple snippets using the method described in Sec. 4.1. For the standard auto-encoder, we treat each snippet as one example, while for the bidirectional recurrent auto-encoder with LSTM cells, we treat the nine-snippet sequence centered on the current snippet as one example. We found that increasing the sequence length has little effect on the performance. For all the experiments conducted in this paper, we use basic auto-encoders with only one hidden layer and linear activation functions. The number of hidden nodes was chosen to be half that of the input layer. Raw features are extracted from the "FC6" layer of the C3D [26] network prior to PCA dimensionality reduction, which maintains 90% of the total energy. We then apply simple mean pooling within each snippet, which provides performance for C3D similar to that of segmented temporal pooling. The performance benefits of employing the 3D spatial-temporal C3D features rather than the 2D single-frame features of Places CNN is shown in Table 2, where the results are obtained using standard auto-encoders trained on each type of feature without applying PCA.

Our results can be reproduced through the following network training parameters which were set without careful tweaking: learning rate of 0.01, weight decay of 0.0005, and momentum of 0.9. We set the maximum epoch number (Γ) to 100, and let λ shrink from 2 to 0.25.

For evaluation, we use the same metric as in [24]. For each video, we sort the snippets in ascending order based on their corresponding reconstruction errors. We then compute the hit rate of the top K snippets with respect to their ground-truth. Finally this number is averaged across the entire testing set to obtain the mean average precision (mAP).

s	e	mAP		
		freeride	skiing	skydiving
0.5	0.25	0.260	0.429	0.330
	0.5	0.277	0.443	0.356
1	0.25	0.278	0.447	0.355
	0.5	0.274	0.423	0.361
	1	0.264	0.423	0.336
2	0.25	0.274	0.437	0.362
	0.5	0.286	0.476	0.373
	1	0.289	0.476	0.360
	2	0.283	0.453	0.366

Table 3. Results with different shrinking exponential parameters. λ shrinks linearly from s to e. Due to limited space, we only show results for three domain categories.

5.3. Results and Discussion

We compare our robust recurrent auto-encoder (RRAE) to other unsupervised alternatives and to the supervised learning technique of [24]. Before that, we examine the effect of different parameters for the shrinking exponential loss.

5.3.1 Effect of Shrinking Exponential Loss

As discussed in Sec. 4.2, using a shrinking exponential loss during training helps to reduce the influence of outliers compared with the standard fixed loss. This is validated in Table 4 by comparing the standard auto-encoder (AE) with its robust version based on the new shrinking exponential loss (robust AE). Intuitively, by gradually changing the shape of loss functions through shrinking λ, the gradients of non-highlight snippets become relatively small so that their influence on network training is gradually reduced. As shown in Table 3, shrinking λ consistently matches or surpasses $s = e = 1$ (the standard fixed squared loss). We also observed in this study that shrinking to small values (e.g., $\lambda = 0.125$) may be unfavorable in some cases as this ends up ignoring too many examples, including inliers. Although carefully tweaking the shrinking range can improve results, we found that going from $\lambda > 1$ to $\lambda < 1$ generally works well for the domain categories we examined. Nevertheless, design of a more optimal shrinking scheme would be an interesting direction for future work.

5.3.2 Unsupervised Learning Comparisons

There exist other unsupervised learning techniques that could be applied to this problem. In addition to the standard auto-encoder (AE), two frequently used methods for anomaly detection and outlier removal are Principal Components Analysis (PCA) and One-class Support Vector Machines (OCSVM) [22]. For PCA, we project the original d dimensional input vector into a d' dimensional subspace, with $d' < d$ as in the auto-encoder. Then snippets with small PCA reconstruction error are taken as highlights. For one-class SVM, we use the LibSVM implementation where its parameters γ (RBF kernel width) and ν (for controlling the outlier ratio) are chosen using a simple line search based on testing error. For the different domain classes, we found that the optimal ν lies in the range of $[0.5, 0.9]$, while γ lies in $[1, 10]$.

Comparisons among these methods are presented in Table 4. Our robust recurrent auto-encoder consistently outperforms AE, PCA and OCSVM on all the domain categories. Although OCSVM works better than PCA, it requires much more computation because of its nonlinear kernel. The table additionally includes comparisons to partial versions of our techniques, namely recurrent AE without the shrinking exponential loss, and non-recurrent AE but with the shrinking loss (denoted as Robust AE). From these results, we can see that when the standard auto-encoder is equipped with LSTM cells (recurrent AE), the performance is boosted by more than 10%, from 0.371 to 0.410. This indicates the importance of modeling the temporal contextual structure of video highlights.

Some of our detection results in different video domains are illustrated in Figure 5. The blue bars represent reconstruction error, with smaller values having a higher probability of being a highlight snippet.

5.3.3 Comparison to Supervised Learning

We have also compared our method to the latent ranking SVM technique in [24], using their YouTube dataset and their mAP evaluation metric. We note that our method is at a significant disadvantage in this comparison, as our system is trained using only the edited videos in the dataset, in contrast to [24] which also utilizes the unedited counterparts. In addition, as the number of edited videos in this training set is relatively small, there is a risk of over-fitting as our system is primarily designed to leverage large-scale web data. The results, listed in Table 5, show that even though the supervised method benefits from major advantages in this comparison, the performance gap on this testing set is small for *dog, gym, parkour* and *surfing*. Moreover, as many of the training and testing video clips are from the same raw videos, it is particularly hard in this case for an unsupervised method such as ours to obtain good results relative to a supervised method trained on pre- and post-edit pairs.

We also examined latent ranking SVM trained on their own data with C3D features, but applied to our testing set. The results are shown in first column of Table 4 on domain categories that are shared by the two datasets. It can be seen that LRSVM does not perform as well as our RRAE. A possible explanation is that the supervised learning has a high risk of overfitting due to the limited training data, and as a result it may not generalize well to large-scale testing

	LRSVM [24]	PCA	OCSVM	AE	Robust AE	Recurrent AE	RRAE
freeride	*	0.235	0.258	0.268	0.277	0.277	**0.288**
parkour	0.246	0.377	0.445	0.507	0.508	0.618	**0.675**
skating	0.330	0.251	0.297	0.308	0.306	0.322	**0.332**
skiing	0.337	0.388	0.412	0.428	0.472	0.478	**0.485**
skydiving	*	0.376	0.332	0.335	0.364	0.338	**0.390**
surfing	0.564	0.525	0.484	0.494	0.534	0.565	**0.582**
swimming	*	0.274	0.238	0.255	0.277	0.275	**0.283**
mAP		0.347	0.352	0.371	0.391	0.410	**0.434**

Table 4. Performance results of our methods and several baseline methods, all using C3D features [26]. The dimensionality of C3D features is reduced from 4096 by a domain specific PCA that keeps 90% of the total energy.

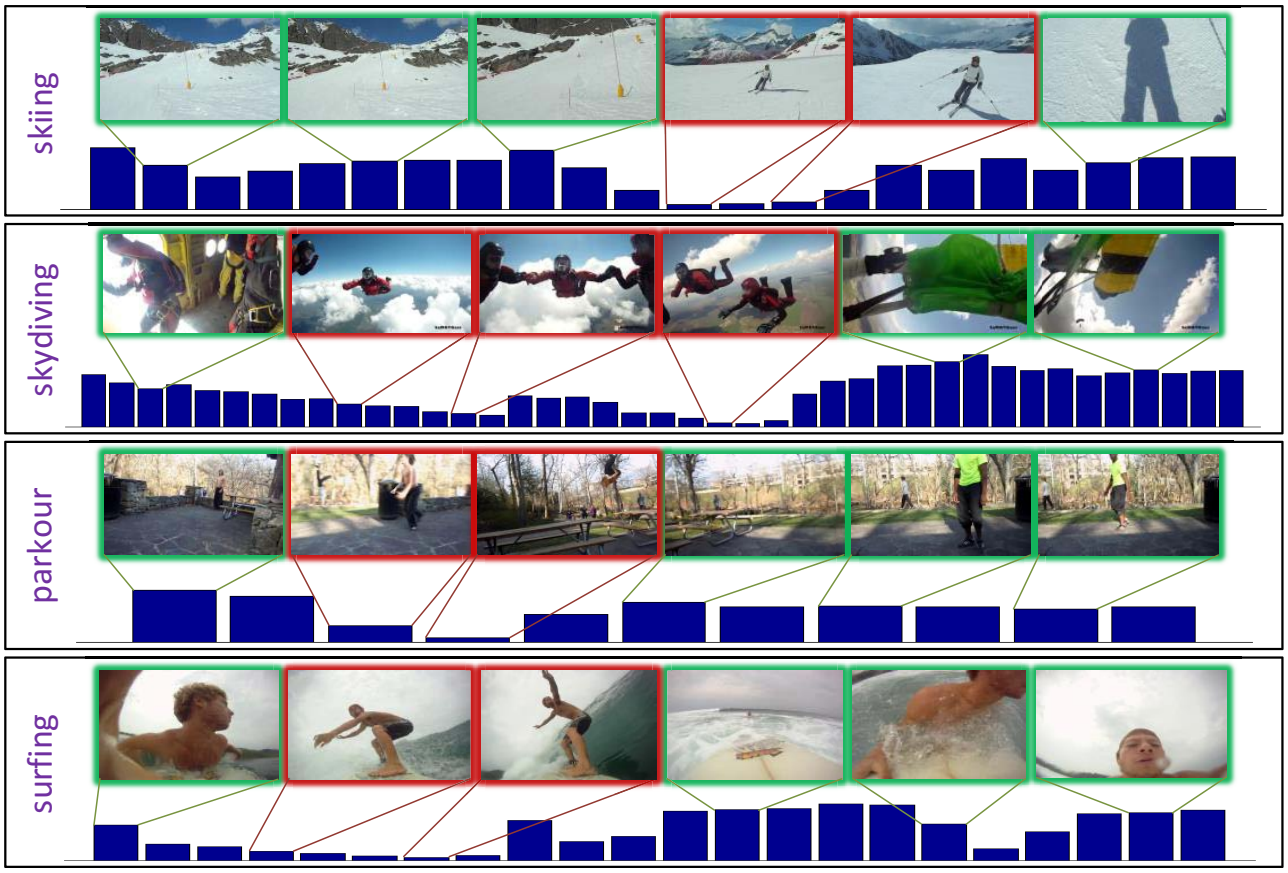

Figure 5. Highlight detection results in different video domains. The blue bar represents reconstruction error, where highlights tend to have smaller errors than non-highlight snippets. The red borders indicate snippets detected as highlights.

	Supervised[24]	*RRAE*
dog	0.60	0.49
gymnastics	0.41	0.35
parkour	0.61	0.50
skating	0.62	0.25
skiing	0.36	0.22
surfing	0.61	0.49

Table 5. mAP comparison to [24] on the YouTube dataset.

sets.

6. Conclusion

We presented a scalable highlight extraction method based on unsupervised learning. Our technique relies on an improved auto-encoder with two significant modifications: a novel shrinking exponential loss which reduces sensitivity to noisy training data crawled from the web, and a recurrent auto-encoder configuration with LSTM cells. Generalizing this technique to other video processing problems would be a potential avenue for future work.

References

[1] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014. 4

[2] 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 27*, pages 2069–2077. 2014. 2, 3

[3] A. Graves and N. Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In T. Jebara and E. P. Xing, editors, *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1764–1772. JMLR Workshop and Conference Proceedings, 2014. 2, 5

[4] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In *ECCV*, 2014. 2, 3, 6

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. 1, 2, 5

[6] N. Japkowicz, C. Myers, and M. Gluck. A novelty detection approach to classification. In *In Proceedings of the Fourteenth Joint Conference on Artificial Intelligence*, pages 518–523, 1995. 3

[7] H.-W. Kang, X.-Q. Chen, Y. Matsushita, and X. Tang. Space-time video montage. CVPR '06, pages 1331–1338, 2006. 3

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, pages 1725–1732, 2014. 4

[9] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1346–1353, June 2012. 3

[10] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. ECCV '02, pages 403–417, 2002. 3

[11] W. Liu, G. Hua, and J. Smith. Unsupervised one-class learning for automatic outlier removal. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3826–3833, June 2014. 3

[12] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *CVPR, 2013 IEEE Conference on*, pages 2714–2721, June 2013. 3

[13] Y.-F. Ma, X.-S. Hua, L. Lu, and H.-J. Zhang. A generic framework of user attention model and its application in video summarization. *Multimedia, IEEE Transactions on*, 7(5):907–919, Oct 2005. 3

[14] L. M. Manevitz and M. Yousef. One-class svms for document classification. *J. Mach. Learn. Res.*, 2:139–154, Mar. 2002. 3

[15] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller. A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 1996–2000, April 2015. 3

[16] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015. 5

[17] C.-W. Ngo, Y.-F. Ma, and H.-J. Zhang. Video summarization and scene detection by graph modeling. *IEEE Trans. Cir. and Sys. for Video Technol.*, 15(2):296–305, Feb. 2005. 3

[18] I. Otsuka, K. Nakane, A. Divakaran, K. Hatanaka, and M. Ogawa. A highlight scene detection and video summarization system using audio feature for a personal video recorder. *IEEE Trans. on Consum. Electron.*, 51(1):112–116, Feb. 2005. 1, 2

[19] D. Potapov, M. Douze, Z. Harchaoui, and C. Schmid. Category-specific video summarization. In *Computer Vision - ECCV 2014,Part VI*, pages 540–555, 2014. 2, 3, 4

[20] Y. Rui, A. Gupta, and A. Acero. Automatically extracting highlights for tv baseball programs. MULTIMEDIA '00, pages 105–115, 2000. 1, 2

[21] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. 1986. 3

[22] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001. 7

[23] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, pages 568–576. 2014. 4

[24] M. Sun, A. Farhadi, and S. Seitz. Ranking domain-specific highlights by analyzing edited videos. In *Computer Vision - ECCV 2014,Part I*, pages 787–802, 2014. 1, 2, 6, 7, 8

[25] X. Tong, Q. Liu, Y. Zhang, and H. Lu. Highlight ranking for sports video browsing. MULTIMEDIA '05, pages 519–522, 2005. 1, 2

[26] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. C3D: generic features for video analysis. *CoRR*, 2014. 3, 4, 6, 8

[27] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *ACM Trans. Multimedia Comput. Commun. Appl.*, 3(1), Feb. 2007. 2

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. ICML '08, pages 1096–1103, 2008. 2

[29] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *IJCV*, 2013. 2, 4

[30] H. Xu, C. Caramanis, and S. Mannor. Outlier-robust PCA: The high-dimensional case. *Information Theory, IEEE Transactions on*, 59(1):546–572, Jan 2013. 3

[31] B. Zhang, W. Dou, and L. Chen. Combining short and long term audio features for tv sports highlight detection. ECIR'06, pages 472–475, 2006. 1, 2

[32] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos. In *CVPR'14*, pages 2513–2520, 2014. 3

[33] B. Zhou, J. Xiao, A. Lapedriza, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 4