A Practical Online Framework for Extracting Running Video Summaries under a Fixed Memory Budget

Chandrashekhar Lavania* Kai Wei† Rishabh Iyer‡ Jeff Bilmes §

Abstract

We study the problem of summarizing a video stream (potentially of unbounded duration) on the fly, where the summarization system must operate under a fixed memory budget and should produce an appropriate summary of its past at each time step. This problem is motivated by applications that have access to only limited memory and compute resources (e.g., sensor networks, smart devices and phones). We approach this problem as constrained streaming maximization of a natural submodular objective function. In particular, we propose a novel feature-based submodular function for the summarization objective — this function is instantiated by deep-model-learnt features. We solve the constrained streaming maximization problem with an algorithm that abides by the memory budget. Our streaming algorithm, is unique in that it uses both an "adding gain" (to determine something new) and a "swapping gain" (to determine something better) relative to a current summary. Based on a time dependent F-measure method to gauge the performance of streaming summarization techniques, we demonstrate that our approach provides significant improvement over a number of state-of-the-art baseline methods that utilize comparable resources on both the TVSum50 and SumMe data sets.

1 Introduction

Many of today's intelligent systems regularly generate vast amounts of video data. This data can be either consumer, surveillance, promotional, or commercial videos, or any form of a live internet video stream. It is not feasible for a human to peruse these videos to gain a deep and complete understanding of their content, as they are typically long and redundant. For example, in surveillance settings, most of the video is repetitive, uninformative, and even when motion-triggered, plagued with false positives. This characteristic makes it easy to miss new interesting data when it does finally arrive. One way to address this problem is to summarize the video content, and hence create an interesting and insightful subset of video "snippets" (i.e., short video segments) for presentation to a human. While summaries can be hand crafted by a human curator, that process itself is time consuming, error prone, and costly given the ever-increasing torrent of video data. It is of great interest, therefore, to develop practical methods for automatic video summarization (i.e., to produce summaries that are informative, representative, and much shorter) and that also perform well using standard evaluation methods. Indeed, this problem has gained significant traction in recent years [29].

In general, existing methods approach video summarization from either an *offline* [11] or an *online* [37] perspective. Offline mechanisms require the knowledge of and access to the entire video stream to generate a summary. Such methods, however, require storing the entire video simultaneously and, therefore, are resource intensive and/or impractical (e.g., for unboundedly long video streams).

Online, or streaming, video summarization methods are an alternative to the above. Given a video stream, an online summarization algorithm produces a summary on the fly and at any time as the data stream elements arrive, without using any future information. Such methods can be made to require limited memory since they keep only a small portion of the past video stream (or information thereabout) in memory. Since online methods are computationally cheaper than their batch counterparts, this setting is of particular interest when batch processing a video is too resource consuming on a device, when an application requires access to the historical summary at any given time, or for unboundedly long video streams.

We are interested in a restricted scenario for online video summarization, where the system has only a fixed (constant) memory budget for retaining the video history. We also wish for a running summary, where we define a "running summary" as one that, at any given point in time, is representative and informative about the observed video stream from the beginning of the stream up until the current time. The overall video stream itself can potentially be of unbounded duration. We call this problem *running online video summarization with a fixed memory budget*.

In this work, we define a new algorithm for this purpose that performs well in practice. Our algorithm is unique in that it uses both an "add gain" (to determine something new) and a "swap gain" (to determine something better) relative to a current summary. Our

*University of Washington, Seattle <lavaniac@uw.edu>
†Microsoft <kawe@microsoft.com>
‡UT Dallas <rishabh.iyer@utdallas.edu>
§University of Washington, Seattle <bilmes@uw.edu>

algorithm generalizes earlier work [2] (TS) which uses only a swap gain, and therefore retains TS's theoretical guarantee of 1/4 as a special case. TS, however, does not use an add gain and can therefore easily produce a summary consisting of temporally concentrated events. Indeed, results show that our algorithm significantly outperforms not just TS but many other baselines as well on both the SumMe [11] and TVSum50 [27] data sets, and does this with far fewer computational resources. This renders our approach practical for many real-world applications in video summarization.

Our approach, moreover, addresses problems that naturally occur in systems with limited resources. Examples include sensor networks where computational resources including processing power, memory and network bandwidth are limited and costly. Suppose, for instance, a camera sensor with limited memory is placed in an inaccessible environment, and only periodic communication with a control hub is allowed, due to limited communications bandwidth. A solution is to perform running online video summarization with a fixed memory budget, and then transmit only the running summaries to the control hub every so often (a transmission requiring a fixed-size batch of information). This reduces both the memory utilization at the sensor node and the bandwidth usage for communication with the control node. Our setting is also useful when a user wants a historical summary on demand at any time (e.g., video surveillance applications), not just the end of the stream.

The problem of video summarization can be categorized into either static or dynamic summarization based on the nature of the selected summary [29]. The goal of static summarization is to choose from a video a number of individual images (called key frames) that are collectively representative. A dynamic summarization scenario focuses on creating a dynamic summary of a video that is composed of set of segments of temporally contiguous images. In this paper we call such a segment (sequence of images) a "*video snippet*." Hence, our summaries are not sets of frames but sets of snippets.

2 Related Work

In the literature, static summarization has often been approached in the offline setting [7, 18, 10]. Recently, Zhang et al [35] used LSTMs to design a supervised approach for selection of key frames or key sub-shots. Mahassen et al [22] took the unsupervised approach and employed deep adversarial LSTM networks for video summarization.

Dynamic video summarization, on the other hand, provides a better viewing experience since each summary snippet is itself a short video segment thereby yielding true short-interval temporal information. Several mechanisms have been explored for generating dynamic summaries. In Gygli et al. [11], Sun et al. [28], and Herranz and Martinez [13] video snippets are selected according to their quality score measuring various aspects of a video (e.g., interestingness and representativeness). More recently, Yao et al. [33] explore summarization of first-person videos through highlight detection. They split their video segment into spatial and temporal streams. Each stream is fed into a deep convolutional neural network for highlight prediction. The outputs of these networks are fused to generate the highlight score of a video segment. Zhao et al. [36] focus on first extracting shots from a video in an adaptive manner using a bidirectional LSTM, and then using a second bi-directional LSTM to assign probabilities to each shot to decide its inclusion in a summary. Kanehira et al. [17] explore video summarization in the context of view-point extracted from multiple videos. They focus on summarizing multiple groups of videos based on the view-point extracted from each group. Jin et al. [16] present a mechanism where the user can interact with the summarizer and aid in producing summaries of different sizes from a given video. It augments an offline frame analysis with user input to fast-forward segments of the video. Cai et al. [3] use a VAE pre-trained on web videos in conjunction with an encoder-attention-decoder setup to generate summaries. Their technique is also explored in the offline setting. Rochan and Wang [25] learn a mapping from a set of videos to a set of summaries where there is no direct correspondence between the video set and the summary set. Their aim is to learn a mapping such that the summaries generated using it have the same distribution as the summary set used in training. Yuan et al. [34] propose an unsupervised framework that learns to generate a summary using a bidirectional LSTM as a summary selector and GAN based evaluator.

Video summarization has also been examined from an online perspective. For instance, Zhao and Xing [37] select video snippets on the fly based on its reconstruction error from a dictionary, which is generated by the snippets that are already in the summary. Iparraguirre and Delrieux [14] propose to summarize a video stream in an online fashion by identifying key frames, from which video snippets in the summary are constructed. In general, these techniques do not directly fall within our fixed memory budget setting, since they do not constrain the size of their summaries. The closest work to our setting is by Mei et al. [23], which again utilizes a reconstruction-error based summarization technique but abides by a fixed summary budget constraint.

Recently, techniques based on submodular opti-

mization have been applied to video summarization. For example, Gygli et al. [12] learn a mixture of components to model the objective for video summarization in the offline setting. Xu et al. [32] formulate summarization of egocentric videos as constrained maximization of an objective function. Again their approach is in the offline setting. Chakraborty et al. [4] consider adaptive key frame selection for video summarization. They formulate the problem as unconstrained maximization, where the objective is defined on a complete similarity graph, and thus the method is inherently offline. Li et al. [21] propose a framework for summarizing both raw and edited videos. They create a submodular scoring function for a set as a weighted mixture of importance, representativeness, diversity and "storyness" of that set with respect to a video. They use a supervised max-margin approach to learn the mixture weights and use an accelerated greedy algorithm to produce their summary. Their approach however is again offline. On the other hand, Elhamifar and Kalusza [9], use an online summarization approach. They focus, however, on unconstrained summarization and once again are not directly comparable to our approach.

Our contributions: In this paper we provide a submodular framework for the problem of running online video summarization with a fixed and limited memory budget. We utilize a novel feature-based submodular function as a summarization objective. Instantiated by deep features, this function naturally models the notion of diversity and informativeness in the online summarization scenario. We optimize the proposed objective by a streaming algorithm that complies with the memory budget at every time step. To assess the quality of a streaming summarization technique, we employ a time dependent F-measure based evaluation method [10, 11, 27, 12, 32, 35, 22]. We empirically demonstrate the efficacy of the proposed summarization framework on the SumMe [11] and TVSum50 [27] data sets. Our approach significantly outperforms a number of baseline methods that employ similar resources.

3 The Summarization Framework

In this section, we first formulate the problem as constrained submodular maximization. We then define our objective that particularly suits this task, and can be optimized through our online procedure.

3.1 Problem Formulation: Let $V = \{v_1, v_2, \dots\}$ be a video stream (potentially of unbounded duration) with v_t as the constituent snippet for time step t. We denote by $V_t = \{v_1, v_2, \dots, v_t\}$ the set of video snippets that have been observed until time step t. Given a memory budget constraint K, for each time step t the goal is then to produce a summary $S_t \subseteq V_t$ of size no larger than K (i.e., $|S_t| \leq K$) such that given a representation criterion, S_t is informative, diverse, and can adequately represent V_t. In our approach we are interested in online summarization and generation of running summaries (of potentially never ending video streams) in a fixed memory budget setting. Therefore, we attempt to generate good summaries on average based upon a desired scoring function. More formally, let a set function $f : 2^V \to \mathbb{R}$ represent the quality of any given subset of V. The problem is equivalent to finding S_t such that the following is achieved for all time t:

$$\max_{S_t \subseteq V_t, |S_t| \leq K} \sum_{\tau=1}^{t} f(S_\tau). \tag{3.1}$$

3.2 Objective Function The choice of the representation criterion $f(\cdot)$ plays an important role in determining the quality of the summary. One possible class of functions that have shown potential in the context of summarization is submodular functions. They are a special class of set functions that satisfy the *diminishing returns* property. In particular, they have the form $f : 2^V \to \mathbb{R}$ and for any $A, B \subseteq V$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. Defining $f(a|A) \triangleq f(a \cup A) - f(A)$, an equivalent definition is $f(a|A) \geq f(a|B) \;\; \forall A \subseteq B \subseteq V,\; a \in V \setminus B$. These functions are natural combinatorial "information" functions that can be applied to arbitrary data objects. They have shown success in modeling diversity, dispersion, span and coverage in prior investigations of video summarization [12, 32, 4]. For instance, the "facility location" function, i.e., $g_{\text{fac}}(S) = \sum_{v \in V} \max_{s \in S} w_{v,s}$, was applied as the objective in Chakraborty et al. [4], where $w_{v,s}$ is the similarity between the video elements (e.g. snippets in our context) v and s. Intuitively, $g_{\text{fac}}(S)$ captures how well the subset S represents the entire data set V. Maximizing $g_{\text{fac}}(\cdot)$ often leads to a collection of representative items.

The drawback of this function is that: 1) it is defined on a pairwise similarity graph, which requires both $O(|V|^2)$ memory and compute; 2) evaluating the function requires access to the entire data V. Therefore the usage of this function in our context (running online summarization) is not scalable. As an alternative, we utilize a function taking the following form:

$$g^{\max}_{\text{fea}}(S) = \sum_{u \in U} w_u \max_{s \in S} m_u(s), \tag{3.2}$$

where U is a set of features, $w_u \geq 0$ is a weight associated with feature $u \in U$ and $m_u(S) = \sum_{s \in S} m_u(s)$ with $m_u(s)$ gauges the degree to which video snippet s possesses feature u. Maximizing $g^{\max}_{\text{fea}}(\cdot)$ leads to di-

verse coverage over the whole feature set U in the selected summary, with the desired coverage of each feature $u \in U$ being controlled by its weight w_u.

In addition to $g^{\max}_{\text{fea}}(\cdot)$, a modular set function $q : V \to \mathbb{R}$ that assigns a quality score for each item $v \in V$ can be added to the objective for influencing the system. The quality function is dependent upon the environment in which the video summarization system is deployed. This can, for instance, be based on a variety of qualities, such as faces, aesthetics, specific places, presence of specific people or objects and many more. Our modular quality function has the following form: $q(S) = \sum_{s \in S} q(s)$, where $q(s)$ is the quality score for the video snippet s based on the desired property. In this work, we explore a simple variant based on a face detection procedure applied to frames in the video. Hence, the quality function encourages the system to select video snippets that contain human faces. We use this quality function as a proof of concept. Systems that are deployed in the field can use much more sophisticated tools. These tools will fit in our framework as long as they produce a modular score.

Furthermore, over a long duration, it may be desirable for a summary to be diverse in time. In this work we utilize a function to encourage time diversity: we first divide the whole video V into "p" (independent of the size of V) sized bins, where the i^{th} bin C_i consists of a set of snippets $\{v_j\}_{j=(i-1)p+1}^{ip}$, i.e., C_i consists of a sequence of snippets that span from the time step $(i-1)p+1$ to ip, and any pair of bins are disjoint, i.e., $C_i \cap C_j = \emptyset \ \ \forall\, i \neq j$. Let n be the total number of bins in the whole video V, we then define our time diversity function $h : 2^V \to \mathbb{N}$ as follows:

$$h(S) = \sum_{i=1}^{n} \min\{|S \cap C_i|, 1\} \tag{3.3}$$

This function evaluates any subset $S \subseteq V$ as the number of bins that are spanned by this set. Note that one does not need the knowledge of the entire data V (or its length) to evaluate this function in the context of online summarization, since, at any given time t, the summary $S_t \subseteq V_t$ does not span any bin corresponding to the future time steps, i.e., it holds that $\min\{|S \cap C_i|, 1\} = 0$ for all $i > \lceil \frac{t}{p} \rceil$. The time diversity function $h(\cdot)$ evaluates a summary higher if the constituent snippets span higher number of bins.

Given the aforementioned three components, the overall objective is defined in the following form:

$$f(S) = g^{\max}_{\text{fea}}(S) + \lambda q(S) + \mu h(S), \tag{3.4}$$

where $\lambda \geq 0$ and $\mu \geq 0$ are the weights of the quality function $q(\cdot)$ and the time diversity function $h(\cdot)$, respectively.

Algorithm 1

Input: f, δ, α, L, K, C, and $V = \{v_1, v_2, \dots\}$.
Output: A running summary S_t for every time step t.
Initialize: $S_0 = \emptyset$, $c_0 = 0$.
for $t = 1, \dots$ **do**
  $g_t \leftarrow f(S_{t-1} \cup v_t) - f(S_{t-1})$
  if $|S_{t-1}| < K$ **then**
    if $g_t \geq \alpha$ **then**
      $S_t \leftarrow S_{t-1} \cup v_t$
    else
      $S_t \leftarrow S_{t-1}$
    end if
  else
    $c_t \leftarrow \frac{1}{L} \sum_{\tau=t-L}^{t-1} g_\tau$; $\gamma_t \leftarrow c_t + \delta$
    $\hat{s} \in \operatorname{argmax}_{s \in S_{t-1}} f(S_{t-1} \cup \{v_t\} \setminus \{s\}) - f(S_{t-1})$
    $\hat{g}_t \leftarrow f(S_{t-1} \cup v_t \setminus \hat{s}) - f(S_{t-1})$
    if $\hat{g}_t \geq C \frac{f(S_{t-1})}{K}$ or $g_t \geq \gamma_t$ **then**
      $S_t \leftarrow (S_{t-1} \cup \{v_t\}) \setminus \{\hat{s}\}$
    else
      $S_t \leftarrow S_{t-1}$
    end if
  end if
end for

3.3 Generation of Running Summaries The problem of streaming video summarization in a fixed memory budget along with the generation of running summaries at each time step has several restrictions. A procedure that addresses this problem must produce summaries that do not exceed the memory budget (i.e., no extra storage or memory may be used for additional snippets). Hence, an incoming snippet that is not introduced to the summary then and there is considered lost. Also, the procedure must assume that the stream duration is unknown. Lastly, given a submodular objective $f(\cdot)$, a streaming procedure needs to be defined that utilizes $f(\cdot)$ to generate the running summaries under these constraints.

As shown in the seminal paper by Nemhauser et al. [24], the problem defined in Equation 3.1 can be efficiently and near-optimally solved with a constant approximation guarantee of $(1 - \frac{1}{e})$ using a greedy algorithm in the offline setting. Recently, a number of streaming algorithms have been proposed for solving this problem in the online setting. In particular, Badanidiyuru et al. [1] present an efficient algorithm that optimizes Problem 3.1 on the fly while achieving a constant approximation guarantee of $(\frac{1}{2} - \epsilon)$. Unfortunately their approach requires $O(\frac{K \log K}{\epsilon})$ memory where K is the budget of the summary. Hence it is not ideal in our restricted scenario, where we require that

the procedure store no more than the specific given value K of items at any given time (despite this, we do evaluate [1] below and find it does not perform as well as our streaming procedure). Buchbinder et al. [2] gives a swapping-based streaming algorithm that outputs a near-optimal running summary at every time step while abiding by a fixed budget K. Given a stream of data items, this algorithm unconditionally selects the first K items into the summary. Afterwards, the algorithm maintains a summary S_t of size K at every time step t, and swaps the new item v_{t+1} with an existing item in the summary if the maximum gain in the objective is beyond a threshold. Buchbinder et al. [2] suggest setting the threshold for each given time t as $C\frac{f(S_t)}{K}$, where C is a constant. Moreover, they show that such a scheme produces a near-optimal running summary S_t for any step t as follows: $f(S_t) \geq \frac{C}{(C+1)^2}\max_{|S|\leq K, S\subseteq V_t} f(S)$. Note that the approximation factor is maximized as $1/4$ when $C = 1$. We call this algorithm "Threshold and Swapping" based streaming optimization (TS).

In this work, we introduce a new method better suited to our task. The modified algorithm (Alg. 1) consists of two stages: (1) adding stage ($|S_t| < K$), and (2) swapping stage ($|S_t| = K$).[1] In the adding stage, the procedure examines each incoming snippet v_t and makes a decision about its inclusion in the summary by analyzing the gain of adding v_t to S_{t-1}. The snippet is added to the summary if the adding gain $g_t = f(v_t|S_{t-1})$ is no less than the threshold α. The setting where $\alpha > 0$, prevents the unconditional selection of the first K video snippets in the data stream if the beginning of the video stream is highly redundant. The adding stage finishes when the summary size reaches the budget K. Note that setting $\alpha = 0$ in our modified variant reduces to unconditional selection of the first K items as is done in TS. Hence, the add stage ensures that the initial K snippets are neither arbitrary nor redundant — filling up the initial summary budget using arbitrary or redundant snippet, as in TS, is wasteful (in practice) and can take some time to recover from, especially when one wants a running summary.

In the swapping stage, a new snippet is added to the summary by swapping it with an existing snippet. Given a time step t, we denote $\hat{g}_t$ as the maximum gain of swapping the incoming element v_t with an existing item $\hat{s} \in S_{t-1}$. The new item v_t is swapped in if either the maximum swapping gain $\hat{g}_t$ exceeds the threshold $C\frac{f(S_{t-1})}{K}$, or (and our novel contribution) the adding gain g_t exceeds another threshold γ_t. In either case, the new element v_t is swapped with the item $\hat{s} \in S_{t-1}$ that provides the maximum swapping gain. A large maximum swapping gain $\hat{g}_t$ implies that the benefit of adding the new snippet outweighs the loss of removing an item from the summary.

The adding gain being large enough triggers a swap whenever the new element is highly distinct from any current snippet in S_{t-1}. This is where our algorithm behaves critically differently than TS. When the budget is fixed, there might be no way to improve the objective via a swap (indeed, we already might be at the optimal set of size K). The adding gain trigger does not attempt to improve the objective, rather it triggers a swap to ensure that something new and good is added to the summary, while allowing something old, or at least not too good, to be removed. This places a preference on novelty at the potential expense of the objective value. The intuition for this is that with a running summary, we wish to avoid a situation where a summary, being highly scored by objective, becomes temporally stale. For example, it can be that the first K elements are optimal, but in a running summarization setting, the first K elements might not make a good summary for all time regardless of how highly scored they are. While the time diversity component of the objective keeps this from happening to some extent, the adding gain further facilitates good quality running summaries. Our evaluation below (Section 4) considers the case both without the time diversity component and without the adding gain, and finds that both are necessary for optimal performance.

The tradeoff between the swapping gain vs the adding gain that triggers a swap is determined by the adaptive threshold γ_t, which is computed based on the moving average of g_t over the past L time steps. This allows the thresholds to adjust automatically and smoothly based on the video stream. The swapping stage in the modified variant reduces to TS when $\gamma_t = \infty$ (or equivalently, $\delta = \infty$), in fact we have:

PROPOSITION 3.1. *For $\alpha = 0$ and $\delta = \infty$, Algorithm 1 will also generate S_t such that $f(S_t) \geq \frac{C}{(C+1)^2}\max_{|S|\leq K, S\subseteq V_t} f(S)$. The approximation factor is maximized as $\frac{1}{4}$ for $C = 1$.*

The efficacy of such modification in TS at the swapping stage is empirically validated in Section 4, where using the add gain yields significant improvements.

4 Experimental Setup

4.1 The Data Set We compare the performance of various summarization methods on the videos from SumMe [11] and TVSum50 [27] data sets. The SumMe data set consists of 25 videos. Each video has a set of human generated summaries (the ground truth) from multiple users. Each summary consists of a set of

[1] A brief demonstration is at https://youtu.be/lTzLWGcb8Mg.

variable length video snippets. A detailed description of the data is given in Gygli et al. [11]. The TVSum50 data set contains 50 videos in 10 categories. The videos were collected from YouTube using the category as the search query. Each video has a set of associated ground truth summaries. A detailed description of the data set is provided in Song et al. [27].

4.2 Instantiation of Objective Function The features that are used to instantiate the objective $f(\cdot)$ play an important role in identifying the focus of the summarization process. In this work we instantiate $g_{\text{fea}}^{\max}(\cdot)$ on a set U of deep features, which is generated using a Deep Neural Network (DNN).

Previous work has shown that DNN features perform well in a number of computer vision tasks [20, 19]. Moreover, Donahue et al. [8] show that a network similar to the one proposed by Krizhevsky et al. [19], can be used as black box generic feature generator for a diverse set of tasks. To instantiate $g_{\text{fea}}^{\max}(\cdot)$ we generate a 100 dimensional feature representation for each video snippet through a three-step process.

First we train Caffe's [15] "AlexNet" (based on the network in Krizhevsky et al [19]) on the ILSVRC2012 (ImageNet) data set [26]. The trained model can then produce a 1000 dimensional first level feature representation for a video image. These features correspond to object classes and are useful from a summarization perspective. However, since the model was trained on ILSVRC2012 data, these features are correlated due to the similarity between object classes in the data set. Therefore, as the second step we utilize a fully-connected DNN model that has a bottleneck structure to refine the pre-softmax first level features into a 100 dimensional representation.

We design an autoencoder style DNN model with 5 layers of hidden nodes, 1 layer of input nodes, and 1 layer of output nodes. We define the bottleneck layer as the third hidden layer with 100 nodes. The model is trained in an unsupervised manner using the ILSVRC2012 data and sigmoid non-linearity. Once the network has been trained, the data from the summarization data sets is passed through the network and the output of the bottleneck layer for each image (video frame) is scaled and transformed by application of the sigmoid non-linearity before being used as the second level feature representation.

At this stage second level features are available for each constituent frame of a snippet. However, there is still a need for features representing the entire snippet. To this end, we train another autoencoder with a bottleneck structure over the frames of the snippet to generate a feature representation for the entire snippet. For each snippet, the features of the constituent frames are concatenated together and fed into the autoencoder. The autoencoder consists of 7 layers of hidden nodes, 1 layer of input nodes and 1 layer of output nodes. The bottleneck layer is the fourth hidden layer with 100 nodes. The sigmoid non-linearity is used in the autoencoder. Once this autoencoder has been trained, it is used to generate third level 100 dimensional features ($|U| = 100$) for each snippet in the summarization data sets. These third level features are used to instantiate $g_{\text{fea}}^{\max}(\cdot)$.

Let a video snippet v be represented as $r(v) \in \mathbb{R}_{+}^{100}$. This $r(v)$ corresponds to the third level features described earlier. In the experiments we define $g_{\text{fea}}^{\max}(\cdot)$ with $m_u(v) = r_u(v)$, and $w_u = 1, \forall u \in U$. The quality function $q(\cdot)$ employed in the experiments is defined via a Haar feature-based cascade classifier [30] for face detection. More formally, given a video snippet v, we define $q(v)$ as either 1 or 0, depending on the presence of a face in the video snippet as predicted by the face detector.

It is important to understand that even though deep neural networks were used to generate snippet features, recent techniques for compressing neural networks (such as Chen et al. [6] and Wang et al. [31]) can allow the usage of such neural networks in memory constrained environments making them more practical in our setting.

4.3 Evaluation Mechanism: Running F score It is common to use F-measure based evaluation techniques for judging an automatically generated video summary [11, 27, 12, 32]. Similarly, to test the performance of running summaries, we employ an F-measure based evaluation method that takes the quality of the summary generated for every time step into account, instead of examining only the end summary of a video. Given a video $V = \{v_1, \ldots, v_n\}$ consisting of a set of snippets and a corresponding human generated summary (ground-truth summary), we denote $d(v_i)$ as the number of frames in the video snippet v_i and $c(v_i)$ as the number of intersecting frames between the human summary and the snippet v_i. We then evaluate the quality of the summary at time step t relative to the human generated summary through the F-measure F_t, which is defined as follows: $P_t = \frac{\sum_{v \in S_t} c(v)}{\sum_{v \in S_t} d(v)}$, $R_t = \frac{\sum_{v \in S_t} c(v)}{\sum_{v \in V_t} c(v)}$, $F_t = 2\frac{P_t R_t}{P_t + R_t}$, where P_t and R_t are precision and recall at time step t. In the case of multiple human summaries, these quantities are calculated by averaging the scores obtained using individual human summaries. The performance of the online summarization method for the video V is defined as:

$$\bar{F} = \frac{1}{n} \sum_{t=1}^{n} F_t \tag{4.5}$$

5 The Results

In the experiments we address the following questions: 1) how does the proposed framework perform compared to a wide variety of baseline methods; 2) how does the proposed framework perform with various choices of the hyperparameters; and 3) how does our proposed adding gain in stage two (swapping stage) improve the performance over not using it at all. We answer these questions through experiments on the videos contained in SumMe (SM) [11] and TVSum50 (TVS) [27] data sets.

The core problem being studied in this work is to generate "running summaries" under constraints (such as bounded memory usage). This has practical importance as outlined in Section 1; the online nature is just one aspect of our approach. Most video summarization methods (including online variants) do not focus on this exact problem. Therefore, it is difficult (and arguably unfair) to use them as comparisons. Any procedure comparable to ours should satisfy the following properties: **A**) it can operate in an online single-pass manner; **B**) it can operate using a fixed limited budget for any and all arbitrary stream lengths; **C**) it can generate a running summary conforming to this budget; **D**) the total length of the video stream is unknown or unbounded; **E**) it does not store extra snippets apart from those present in the summary at any given time. In our approach, any snippet not added to the summary is considered lost and is no longer accessible to the summarization procedure. These conditions restrict our options for baseline methods. It is rare for a method to consider generation of such running summaries. Table 1 demonstrates that several contemporary methods do not conform to our setting. There are, however, procedures in this table that do satisfy a majority of the required properties. We use these procedures as our baselines.

Condn	KC	RR	TS	Sieve	Ours	[37]	[9]	[22]	[17]	[36]	[3]
A	✓	✓	✓	✓	✓	✓	✓				
B	✓	✓	✓	✓	✓						
C	✓	✓	✓	✓	✓						
D	✓	✓	✓	✓	✓						
E	✓	✓	✓		✓	✓					

Table 1: The different properties satisfied by baseline methods. A, B, C, D, and E represent the different properties that were mentioned earlier. As can be seen, many of the methods do not satisfy any of the properties at all.

We compare our approach against baseline methods, including the streaming optimization procedure proposed by Doubling K-Centers algorithm [5] (KC) and Resource Restricted Online Minimum Sparse Reconstruction procedure [23] (RR). Furthermore, even though the 'SieveStreaming' procedure [1] uses much more memory than our procedure, it uses a submodular objective and has a mathematical performance guarantee. To achieve a good guarantee, we set low ϵ values for SieveStreaming. We also compare our performance with TS [2]. Both SieveStreaming (Sieve) and TS act as strong baselines for comparison with our approach.

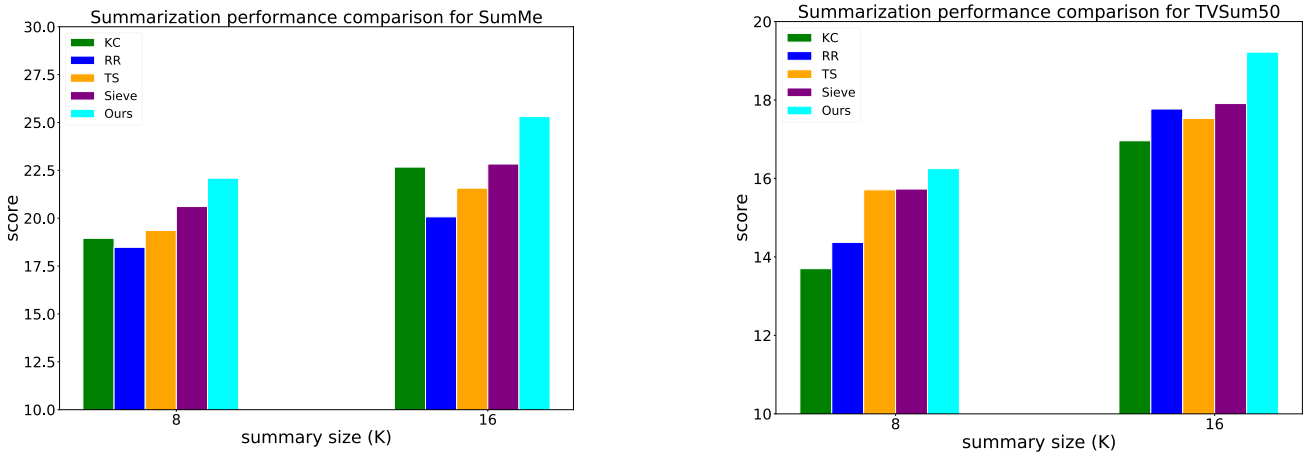
Data Set	K	NA + NT	NA + T	A + NT	A+T
TVS	8	15.79	15.88	16.11	**16.25**
TVS	16	18.70	18.76	19.07	**19.22**
SM	8	21.32	21.38	21.92	**22.09**
SM	16	24.72	24.76	25.11	**25.31**

Table 2: An ablation study depicting the influence of time diversity (T) and adding gain (A) during swapping stage on the performance of our approach. NA and NT correspond to absence of T and A. The score is averaged $100 \times \bar{F}$ (higher is better).

The idea of the core set based KC algorithm is to summarize the video stream on the fly while maintaining the summary such that the minimum distance between any pair of the constituent snippets is sufficiently large. This algorithm operates in a fixed memory budget scenario, and hence it can be compared with our method. The RR procedure also summarizes the video stream without overflowing the budget. We adapt the algorithm to our problem setting: it is instantiated by our deep features and produces summaries of video snippets. This procedure maintains a summary based on the reconstruction error of an incoming snippet.

For each video in SumMe and TVSum50 we fix the size of a video snippet to 2 seconds (as suggested in TVSum50 [27]) and evaluate with the two different budget size. We used a budget size K of both 8 and 16 in the experiments. The averaged $\bar{F}$ (Equation 4.5) for the different methods is shown in Figures 1a and 1b (the scores are multiplied by 100). It can be seen that our procedure clearly outperforms the other baselines suggesting the efficacy of the proposed objective $f(\cdot)$ for modeling the summarization problem. In addition to this, our procedure can perform better than the more memory intensive 'SieveStreaming' [1]. It also shows that our procedure can perform better that TS [2].

In addition to this, through Table 2 it is demonstrated as an ablation study that both time diversity and adding gain at the swapping stage are useful for summarization performance. Furthermore, the second data column of Table 2, as compared to TS in Figures 1a



(a) (b)

Figure 1: Comparison between our framework and several baselines on the (a) SumMe and (b) TVSum50 data sets for different summary budgets (K) using average $100 \times \bar{F}$.

and 1b shows the benefit of the add stage of our algorithm in practice — rather than indiscriminately adding the first K snippets to the summary as in TS, it is beneficial to gradually add snippets at the beginning of the stream depending on if they are novel.

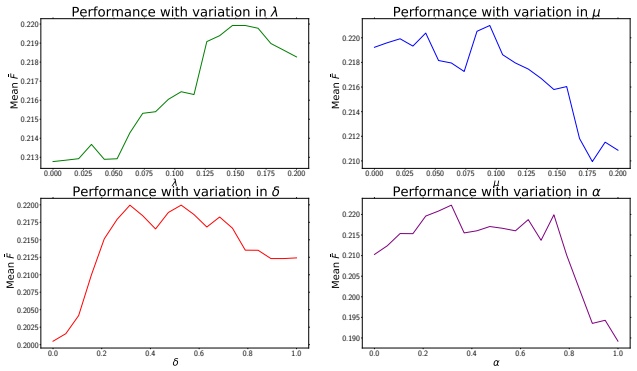


Figure 2: The performance ($\bar{F}$ averaged over all videos) of our approach with varying δ, α, λ and μ on SumMe data set for budget $K = 8$.

Next we examine how the system performance changes with variations of the hyperparameter λ and μ, that define the weights regarding the quality and time diversity functions in the overall objective function. We also explore the influence of the hyperparameters α and δ, that control the threshold for adding a snippet during the add stage, and the influence of the adding gain during the swap stage. The result is shown in Figure 2. It can be observed that appropriate choices of these hyperparameters can have a positive influence on the framework. However, setting the values too large implies excessive emphasis on the associated components leading to deteriorated performance. Overall, a balance of these parameters is needed for optimal performance.

6 Discussions and Conclusions

We present a framework for generating running summaries of video data streams (of potentially unbounded length) on the fly while abiding by a fixed memory budget. We address this problem using a constrained maximization formulation, for which we propose a novel feature-based objective that is utilized in a streaming procedure to generate the running summaries. Furthermore, we show an evaluation scheme for analyzing the performance of streaming methods that can generate running summaries in the context of video summarization. Empirical validation on the SumMe and TVSum50 data sets has shown that the use of our framework for this task has merit.

This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA.

References

[1] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: Massive data summarization on the fly. In *SIGKDD*. ACM, 2014.

[2] N. Buchbinder, M. Feldman, and R. Schwartz. Online submodular maximization with preemption. In *SODA*. ACM-SIAM, 2015.

[3] S. Cai, W. Zuo, L. S. Davis, and L. Zhang. Weakly-supervised video summarization using variational encoder-decoder and web prior. In *ECCV*, pages

184–200, 2018.

[4] S. Chakraborty, O. Tickoo, and R. Iyer. Adaptive keyframe selection for video summarization. In *WACV*. IEEE, 2015.

[5] M. Charikar, C. Chekuri, T. Feder, and R. Motwani. Incremental clustering and dynamic information retrieval. *SIAM Journal on Computing*, 33(6), 2004.

[6] W. Chen, J. Wilson, S. Tyree, K. Weinberger, and Y. Chen. Compressing neural networks with the hashing trick. In *ICML*, pages 2285–2294, 2015.

[7] 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), 2011.

[8] 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*, 2014.

[9] E. Elhamifar and M. C. D. P. Kaluza. Online summarization via submodular and convex optimization. In *CVPR*, pages 1818–1826, 2017.

[10] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *NeurIPS*, 2014.

[11] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool. Creating summaries from user videos. In *ECCV*. Springer, 2014.

[12] M. Gygli, H. Grabner, and L. Van Gool. Video summarization by learning submodular mixtures of objectives. In *CVPR*, pages 3090–3098, 2015.

[13] L. Herranz and J. M. Martínez. An efficient summarization algorithm based on clustering and bitstream extraction. In *ICME*. IEEE, 2009.

[14] J. Iparraguirre and C. Delrieux. Speeded-up video summarization based on local features. In *ISM*. IEEE, 2013.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM Multimedia*, volume 2, 2014.

[16] H. Jin, Y. Song, and K. Yatani. Elasticplay: Interactive video summarization with dynamic time budgets. In *ACM Multimedia*, pages 1164–1172, 2017.

[17] A. Kanehira, L. Van Gool, Y. Ushiku, and T. Harada. Viewpoint-aware video summarization. In *CVPR*, pages 7435–7444, 2018.

[18] G. Kim, L. Sigal, and E. P. Xing. Joint summarization of large-scale collections of web images and videos for storyline reconstruction. In *CVPR*. IEEE, 2014.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012.

[20] C. Lavania, S. Thulasidasan, A. LaMarca, J. Scofield, and J. Bilmes. A weakly supervised activity recognition framework for real-time synthetic biology laboratory assistance. In *Ubicomp*, pages 37–48. ACM, 2016.

[21] 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.

[22] B. Mahasseni, M. Lam, and S. Todorovic. Unsupervised video summarization with adversarial lstm networks. *CVPR*, 2017.

[23] S. Mei, Z. Wang, M. He, and D. Feng. Resource restricted on-line video summarization with minimum sparse reconstruction. In *Picture Coding Symposium (PCS), 2015*. IEEE, 2015.

[24] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical Programming*, 14(1), 1978.

[25] M. Rochan and Y. Wang. Video summarization by learning from unpaired data. In *CVPR*, pages 7902–7911, 2019.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 2014.

[27] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes. Tvsum: Summarizing web videos using titles. In *CVPR*, 2015.

[28] M. Sun, A. Farhadi, and S. Seitz. Ranking domain-specific highlights by analyzing edited videos. In *ECCV*. Springer, 2014.

[29] B. T. Truong and S. Venkatesh. Video abstraction: A systematic review and classification. *TOMM*, 3(1), 2007.

[30] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, volume 1. IEEE, 2001.

[31] S. Wang, H. Cai, J. Bilmes, and W. Noble. Training compressed fully-connected networks with a density-diversity penalty. In *ICLR*, Toulon, France, 2017.

[32] J. Xu, L. Mukherjee, Y. Li, J. Warner, J. M. Rehg, and V. Singh. Gaze-enabled egocentric video summarization via constrained submodular maximization. In *CVPR*, 2015.

[33] T. Yao, T. Mei, and Y. Rui. Highlight detection with pairwise deep ranking for first-person video summarization. In *CVPR*, pages 982–990, 2016.

[34] L. Yuan, F. E. Tay, P. Li, L. Zhou, and J. Feng. Cycle-sum: cycle-consistent adversarial lstm networks for unsupervised video summarization. In *AAAI*, volume 33, pages 9143–9150, 2019.

[35] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. In *ECCV*, pages 766–782. Springer, 2016.

[36] B. Zhao, X. Li, and X. Lu. Hsa-rnn: Hierarchical structure-adaptive rnn for video summarization. In *CVPR*, pages 7405–7414, 2018.

[37] B. Zhao and E. P. Xing. Quasi real-time summarization for consumer videos. In *CVPR*. IEEE, 2014.