Sequence-to-Segments Networks for Detecting Segments in Videos

Zijun Wei*, Boyu Wang*, Minh Hoai, *Member, IEEE,* Jianming Zhang, Xiaohui Shen, Zhe Lin, *Member, IEEE,* Radomír Měch, Dimitris Samaras,

Abstract—Detecting segments of interest from videos is a common problem for many applications. And yet it is a challenging problem as it often requires not only knowledge of individual target segments, but also contextual understanding of the entire video and the relationships between the target segments. To address this problem, we propose the Sequence-to-Segments Network (S^2N), a novel and general end-to-end sequential encoder-decoder architecture. S^2N first encodes the input video into a sequence of hidden states that capture information progressively, as it appears in the video. It then employs the Segment Detection Unit (SDU), a novel decoding architecture, that sequentially detects segments. At each decoding step, the SDU integrates the decoder state and encoder hidden states to detect a target segment. During training, we address the problem of finding the best assignment of predicted segments to ground truth using the Hungarian Matching Algorithm with Lexicographic Cost. Additionally we propose to use the squared Earth Mover's Distance to optimize the localization errors of the segments. We show the state-of-the-art performance of S^2N across numerous tasks, including video highlighting, video summarization, and human action proposal generation.

Index Terms—Segment detection, video analysis, video summarization, video highlighting, video temporal action proposal

1 Introduction

We address the problem of detecting temporal segments of "interest" in videos: as shown in Fig.1, a video is represented as a sequence of input frames, the task is to find a set of consecutive frames of interest. Here, we define "interest" as an abstract concept that denotes the parts of the data that have the highest (application dependent) semantic values. We assume there are training videos with annotated segments of interest (e.g., labeled by humans), and our goal is to train a neural network that can detect the segments of interest in unseen videos. This general problem arises in many situations including temporal event detection [1], [2], [3], video summarization [4], [5], [6], video highlighting [7], [8], discriminative localization [9], [10], [11], and many other research areas such as sentence chunking [12] and gene localization [13]. Each application has its own definition for the segments of interest, and we assume the availablity of annotated training examples. For human action detection, the segments of interest are the ones that correspond to the temporal extents of human actions. For video highlighting or summarization, the segments of interest are the video snippets that are most interesting or best summarize the video.

A typical approach to address this problem is to train a classifier to separate the annotated segments of interest from negative examples. Once trained, the classifier can be used to evaluate individual candidate segments of the input time series in a sliding window approach to identify segments of interest. This approach however has three drawbacks: (i) the computational complexity depends on the number of candidate segments, and this scales quadratically with the length of the time series. More importantly (ii), the independent processing of each segment is suboptimal for many situations because "interest" might be a contextual concept: to detect a set of target segments, not only do we need to evaluate the local content of individual segments, but also their collective relationships and their roles in the global context. As an example, to summarize a video, it is important to know and preserve the gist of the video, and this requires a holistic analysis of the video. Furthermore (iii), the set of selected video snippets in video summarization should not overlap temporally or semantically, and this can only be avoided by collectively evaluating the segments. The second and third drawbacks of the sliding window based classification approaches are commonly addressed by dedicated post-processing steps. However, adding post processing steps complicates the pipeline which cannot be optimized end-to-end.

In this paper we propose the Sequence-to-Segments Network (S^2N), a novel recurrent neural network for analyzing a video to detect temporal segments of interest. Our network is based on the sequential encoder-decoder architecture [14]. The *encoder* network encodes the time series and produces a sequence of hidden states that progressively capture from local to holistic information about the times series. The *decoder* network takes the final state of the encoder network as its starting state and outputs one segment of interest at a time. The state in the decoder will be updated to incorporate what has been already output. This alleviates the need for a post-processing step that may not have access to the time series information.

We introduce a novel architecture for the decoder network, the Segment Detection Unit (SDU), which outputs a segment based on the decoding state and the hidden

- * *Both authors have contributed equally to this work.*
- *Zijun Wei, Boyu Wang, Minh Hoai, Dimitris Samaras are with the Department of Computer Science, Stony Brook University, Stony Brook, NY, 11794.*
- *Jianming Zhang, Zhe Lin and Radomír Měch are with Adobe Research. San Jose, CA, 95110*
- *Xiaohui Shen is with ByteDance AI Lab, Melon Park, CA, 94025*

states of the encoder. The SDU localizes the segment of interest by pointing to the boundaries of the segment, similar to the pointer network [15]. The SDU also outputs a confidence value for the selected segment. The computational complexity of the SDU is linear with respect to the length of the input sequence, which is more efficient than the quadratic complexity of the sliding window approach. Moreover, the whole encoder-decoder pipeline can be optimized end-to-end.

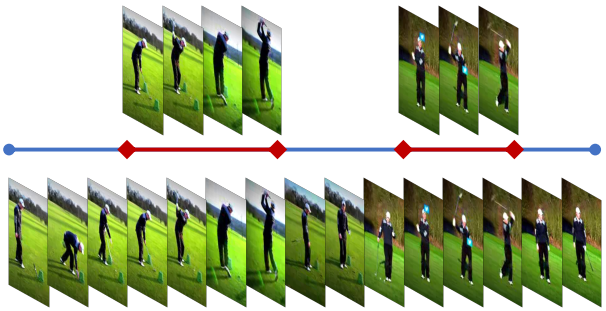


Fig. 1: Detecting segments of interest in videos: given an input sequence, the task is to detect segments of "interest" from the video. The "interest" here is an abstract concept that denotes the segment of the data that have the highest (application dependent) semantic values. Typical applications are video summarization, video highlighting and video temporal action proposal

We learn an S^2N by alternating between two stages: (i) assigning multiple detected segments with ground truth (target) segments, (ii) optimizing S^2N with loss functions on the localization offsets based on the assignment. Inspired by [16], [17], we match the target segments of interest and the sequence of detected segments to have the minimum matching cost which is computed using the Hungarian Matching with a Lexicographic Cost (HMLC). In our experiments, we compared HMLC to its alternatives and observed that S^2N trained with HMLC achieved better performance. For the loss functions, we use the squared Earth Mover's Distance loss to account for the localization error of the boundary pointing modules in the SDU.

To sum up, the major contributions of this paper are:

1) We propose S^2N, a novel network architecture based on SDU for detecting multiple segments of interest in a video;
2) We propose a novel strategy to train S^2N effectively with the HMLC matching algorithm and the squared EMD loss;
3) Through thorough evaluation, we conclude that HMLC and squared EMD are better suited to this problem than more commonly used alternatives;
4) We additionally propose a simple and yet effective beam-search approach to make S^2N generate more diversified proposals, which improves performance.
5) We show that S^2Ns outperform the state-of-the-art methods in three real-world applications with minimum post-processing operations: *i) video highlighting*, *ii) video summarization*, and *iii) human action proposal generation*.

2 Related Work

2.1 Encoder-Decoder RNNs

Recurrent Neural Networks (RNNs) have been the standard method for learning functions defined over sequences from examples sincve the early days of Neural Networks [18]. To further remove the constraint that the number of outputs is dependent on the number of inputs, Sutskever *et al*. [14] proposed the sequence-to-sequence paradigm that first uses one RNN to map an input sequence to a state vector and then applies another RNN to output a sequence with arbitrary length based on the encoded state. Bahdanau *et al*. augmented the decoder by propagating extra contextual information from the input using a content-based attentional mechanism [19], [20]. Vinyals *et al*. [15] modified the attention model to allow the model to directly point to elements in the input sequence, providing a more efficient and accurate model for element localization. These developments have made it possible to apply RNNs to new domains such as language translation [14], [19] and parsing [21], and image and video captioning [22], [23]. However, since standard RNNs are designed to output each time one "token" in the input sequence, they cannot directly handle the segment detection task in which each time a continuous subsequence of the input is selected. Perhaps the most related work to ours is [17] which attempted to train RNNs to label unsegmented sequences directly. But the goal of [17] was classification where the localization information was not required in the output. The proposed S^2N simultaneously detects segments and estimates their confidence scores, thus can be applied to different problems such as temporal action proposal generation and video summarization.

2.2 Video Highlight Detection, Video Summarization & Temporal Action Proposal

Video Highlight Detection Video highlight detection focuses on detecting the temporal clips which contain the most salient event. Early highlight detection works focuses on broadcast sport videos [24], [25], [26]. Given the increase of user uploaded video, recent work focuses on detecting highlights in generic personal videos. [27] proposes to generate the highlighted segment that corresponds the most with the video title. They first extract features from each video clip and train a bidirectional LSTM to predict a highlight score of each clip. Then they simply select the eight consecutive clips with the highest score as the highlighted region. Thus, this approach is only able to generate proposals with fixed length. [28] utilize user preferences to identify highlights in each domain. However, candidate segmentations have to be detected at a pre-processing stage.

Video Summarization Video summarization aims to shorten a video while preserving the important and relevant information it contains. Multiple methods have been proposed recently and, depending on whether they are trained with summaries created by humans, they can be divided into supervised [4], [5], [29], [30] and unsupervised [31], [6], [7], [32] video summarization. Supervised video summarization methods aim to maximize the correspondence between the manually created summarization and the generated summarization. DPP-LSTM [4] utilized sequence-to-sequence

models for this task. [29], [30], [6] use hierarchical LSTMs and attention mechanisms to extract embeddings of the input video. Most unsupervised methods [32], [33], [34] rely on manually designed criteria e.g. how important, representative and diverse are the segments. Recently, [6], [7], [31] generate unsupervised video summarization by outputting a summary which can generate another video that is similar to the original input video.

Video Temporal Action Proposal Temporal action proposal generation aims to generate temporal boundaries of action instances in untrimmed videos. Proposal generation is an important step for action detection and analysis. Earlier works directly use sliding windows as proposals. However, this approach suffers from high computational cost as it scales quadratically with the length of the video. Sparse-prop [35] applies dictionary learning for proposal generation over a large set of candidate proposals. DAPs [36] adopts recurrent networks and a regression branch for temporal localization. Turn [37] uses boundary regression to evaluate candidate proposals. However, these methods [35], [36], [37] only generate proposals with pre-defined durations and intervals, which are not temporally precise and not flexible enough to cover variable temporal durations. Recent work [38], [39]is able to generate proposals with flexible boundaries and durations. TAG [38] adopts the watershed algorithm to generate proposals from an action-ness score, but it only considers each segment independently and does not include global context for proposal generation. BSN [39] contains multiple-stage pipelines to detect and groups boundaries as proposals, thus cannot be optimized end-to-end.

3 S2N Network Architecture

In this section, we will describe the architecture of a Sequence-to-Segments Network (S^{2}N). We first formally state the problem. We then describe the overall S^{2}N architecture and the details of the proposed Segment Detection Unit (SDU), the core component of S^{2}N for localizing a temporal segment of interest.

3.1 Problem Formulation

Let $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M)$ be an input time series of length M, where $\mathbf{x}_m \in \mathcal{R}^d$ is the observation feature vector at time m. Our goal is to train a recurrent network that can localize a set of segments of interest $\mathcal{S} = (S_1, \cdots, S_N)$ from the input time series $\mathbf{X}$. Here, each segment S_n corresponds to a contiguous subsequence of $\mathbf{X}$ and it is parameterized by a tuple of three variables (b_n, d_n, c_n) indicating the beginning position b_n, the ending position d_n, and the estimated interest score c_n. Unlike [40], [41], there are no explicit constraints on the locations and extents of the output segments; they can overlap and their union does not have to cover the entire sequence $\mathbf{X}$. Intuitively, many problems that detect temporal segments in a series such as action detection or video summarization can be formulated this way.

3.2 Model Overview

The proposed S^{2}N is illustrated in Figure 2. S^{2}N is a sequential encoder-decoder with an attentional mechanism [19]. S^{2}N sequentially encodes an input sequence $\mathbf{x}_1, \cdots, \mathbf{x}_M$ and obtains a corresponding sequence of encoding state vectors $\mathbf{e}_1, \cdots, \mathbf{e}_M$; the encoding state vector $\mathbf{e}_m$ essentially contains integrated information from $\mathbf{x}_1$ to $\mathbf{x}_m$ [14], [42]. In the decoding stage at each step the Segment Detection Unit (Section 3.3) outputs a temporal segment of interest S_n.

3.3 Segment Detection Unit (SDU)

A key component of the S^{2}N is the Segment Detection Unit (SDU) for localizing a segment of interest in the decoding stage. As shown in Figure 2, the SDU is composed of four components: a Gated Recurrent Unit (GRU) [43] that updates and communicates states between decoding steps, two pointing modules [15] that point to the beginning and ending positions of the segment, and a score estimator that evaluates the interest score of the segment. Details about these components follow:

GRU for state update. During decoding, at each step given the previous hidden state $\mathbf{h}_{j-1}$ ($\mathbf{h}_0$ is the concatenation of the last hidden state and the memory cell of the encoder), the GRU module updates the current hidden state: $\mathbf{h_j} = \text{GRU}(\mathbf{h}_{j-1}, \mathbf{z})$, where $\mathbf{z}$ is a learned input vector to the GRU and is common to all the decoding steps. For further details about the GRU, please refer to Cho *et al*. [43].

S^{2}N is a general framework and theoretically any RNN architecture, including LSTMs and Depth-Gated Recurrent Neural Networks [44], can be used in place of the GRU. We propose to use the GRU [43] here because it has a simpler architecture and fewer parameters than other RNNs (which means higher training and testing efficiency). We also experimented with LSTMs but did not observe significant difference in terms of model accuracy. This is consistent with prior observations [45] and empirical findings from prior work on deep recurrent models in other domains [43], [46], [47].

Pointing modules for boundary localization. Given the current state $\mathbf{h}_j$ of an SDU, we predict the two boundary positions similar to the pointer networks (Ptr-Net) [15]. To localize the beginning position b_n of a segment at decoding stage n, we use the pointer mechanism as follows:

$$b_n = \underset{i}{\text{argmax}}\, g(\mathbf{h}_n, \mathbf{e}_i), \tag{1}$$

$$g(\mathbf{h}_n, \mathbf{e}_i) = \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{e}_i + \mathbf{W}_2 \mathbf{h}_n). \tag{2}$$

The beginning boundary is determined as the location i of the encoding sequence that has the highest correlation with the current decoding state ($\mathbf{h}_n$). To measure the correlation, we use a non-linear pointing function g. The output of this function depends on the state $\mathbf{h}_n$ of the SDU and the encoding vector $\mathbf{e}_i$ of the encoder component.

Note the difference from original Ptr-Net [15]: the pointer function is defined based on the encoding state vector $\mathbf{e}_i$ instead of the input vector $\mathbf{x}_i$. The encoding state vector $\mathbf{e}_i$ contains richer information than the input vector $\mathbf{x}_i$; $\mathbf{e}_i$ integrates the progression of the input time series up until time i, and this information is crucial for determining the segment boundaries [48]. In the above, $\mathbf{v}$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are learnable parameters of the pointing module that associates the decoding state with the hidden encoding states.

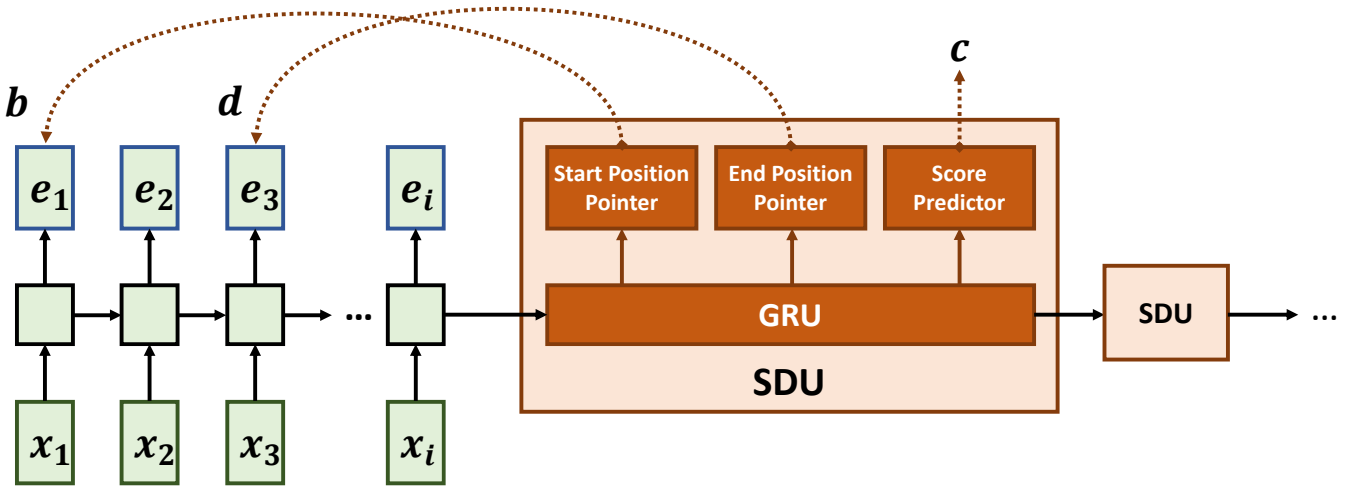


Fig. 2: An Encoder (green) processes the input sequence to create a set of encoding vectors ($\{e_1, e_2, ...e_M\}$). At each decoding step, a Segment Detection Unit (SDU) updates the decoding state with a GRU, and based on the updated state, the SDU points to the beginning (**b**) and ending positions (**d**) with two separate pointing modules and estimates the confidence score (**c**) of the segment.

Similarly, the ending position d_n is determined using another independent Ptr-Net module. Thus, we have two Ptr-Net modules for determining the locations of the beginning and ending positions.

A commonly used alternative for predicting the locations is to use regression (similar to [49], [50]), however, this approach outputs in the $[0, 1]$ range, which does not respect the constraint that the outputs map back exactly to the boundaries and complicates the localization problem. As demonstrated in prior works [51], [15], due to the smoothness of L2 losses, the predicted boundaries are "blurry" and hence difficult to effectively localize the boundaries in longer sequences.

Score predictor. Finally, we attach, f_{score}, a two-layer 1D convolution network on top of the GRU hidden state with a ReLu activation layer in between to predict c_n:

$$c_n = f_{score}(\mathbf{h}_n) \tag{3}$$

Note the c_n is only based on the hidden state and not directly using information from $\mathbf{x}_i$ or $\mathbf{e}_i$, thus this score does not directly evaluate the quality of the segment, but instead predicts whether to accept or discard the output at n. It is possible to incorporate information from $\mathbf{x}_i$ or $\mathbf{e}_i$ into f_{score}, but this will significantly complicate the gradient flow during training.

No terminal output. We do not design a terminal output for S^{2}N as in [19] for two reasons. First, the problem we address is to output a ranked list of temporal segments of interest, which is different from the problem of sequence-to-sequence translation, in which there is a need for a terminal state. Second, by not having a terminal state, S^{2}N can output as many segments as needed, and later the output of the score predictor could be used to select a subset of them, hence allowing flexibility to different needs in real-world problems.

4 S^{2}N Training

The S^{2}Ns can be trained end-to-end. In this section, we first present the overall loss function (Sec. 4.1), we then describe matching the sequence of predicted segments to the set of target segments using the Hungarian Matching with Lexicographic Cost (HMLC, Sec. 4.2), we later analyze HMLC (Sec. 4.3) and compare it to its alternatives (Sec. 4.4) to demonstrate it effectiveness.

4.1 Loss functions

Training an S^{2}N requires a loss function that can measure the discrepancy between an unordered set of ground truth segments $\mathcal{G} = \{G_1, \cdots, G_K\}$ and an ordered sequence of predicted segments $\mathcal{S} = (S_1, \cdots, S_N)$. In other words, when the system is trained we expect the first few predicted segments maximally align with the ground truth. One strategy for matching $\mathcal{G}$ to $\mathcal{S}$ is to use an injective mapping: $f : \{1, \cdots, K\} \to \{1, \cdots, N\}$, where the ground truth instance G_k should be matched to $S_{f(k)}$, and no two ground truth instances should be mapped to the same output segment.

Assume the assignment strategy is known for now, then the loss value for the predicted sequence of segments and the set of ground truth instances is computed as follows:

$$\mathcal{L}(\mathcal{G}, \mathcal{S}, f) = \alpha \sum_{k=1}^{K} \mathcal{L}_{loc}(G_k, S_{f(k)}) + \sum_{n=1}^{N} \mathcal{L}_{conf}(S_n, \delta_n), \tag{4}$$

where δ_n is the $\{0, 1\}$ indicator for S_n depending on whether S_n is matched to a ground truth instance in $\mathcal{G}$. $\mathcal{L}_{loc}$ and $\mathcal{L}_{conf}$ are the loss functions for localization and label assignment, which will be explained below.

Note that $\mathcal{L}_{loc}$ only penalizes K out of the N S_n's, the ones that are matched to the target segments in $\mathcal{G}$. For the remaining $N - K$ unmatched segments, there is no localization penalty.

Loss function for localization. Let us define the loss $\mathcal{L}(G_k, S_{f(k)})$. Our goal is to use a loss function that can

indicate the *level* of discrepancy (localization error) between a predicted segment and a target segment. The surface of the loss function should vary smoothly without any flat regions; flat regions have zero gradients that make gradient-based optimization difficult.

We propose to use a loss function that is based on the Earth Mover's Distance (EMD) between the probability distribution of the predicted boundary and the distribution that represents the ground truth boundary. We now explain how this loss function can be computed for the beginning position b (the loss for the ending position d is computed similarly). Recall from Eq. (2) that we determine the beginning location of a segment as the maximum of a response function: $b = \text{argmax}_i\, g(\mathbf{h}_{f(k)}, \mathbf{e}_i)$, where $\mathbf{h}_{f(k)}$ is the state vector of the SDU. We define the probability of picking i as the boundary point based on the soft-max function

$$Pr(b = i) = \frac{\exp(g(\mathbf{h}_{f(k)}, \mathbf{e}_i))}{\sum_{i=1}^{M} \exp(g(\mathbf{h}_{f(k)}, \mathbf{e}_i))}. \quad (5)$$

Let $\mathbf{p}^*$ be the binary indicator vector for the ground truth location of segment boundary; $p_i^* = 1$ if i is the annotated boundary and 0 otherwise. The EMD loss can be computed based on the differences between the two cumulative distributions:

$$\mathcal{L}_{loc}^{b}(G_k, S_{f(k)}) = \sum_{m=1}^{M} \left(\sum_{i=1}^{m} Pr(b = i) - \sum_{i=1}^{m} p_i^* \right)^2. \quad (6)$$

Here, we use the sum of squared differences in Eq. (6) instead of the sum of absolute differences because the former is easier to optimize with gradient descent [52], [53], [54]. The prediction loss for the ending position is similarly defined and the total localization loss is:

$$\mathcal{L}_{loc}(G_k, S_{f(k)}) = \mathcal{L}_{loc}^{b}(G_k, S_{f(k)}) + \mathcal{L}_{loc}^{d}(G_k, S_{f(k)}). \quad (7)$$

One alternative localization loss function is the cross-entropy loss [15]. However, it is unsuitable for boundary localization because it is insensitive to the amount of localization error: this loss function incurs uniform gradients when the predicted boundary is not exactly the target boundary. Another alternative is to use the Mean Squared Error (L_2) loss, but this loss function is also unsuitable for the same reason.

We will provide an empirical comparison between the proposed squared EMD loss and the L_2 and cross-entropy losses in Section 5.2.

Loss function for confidence score estimation. The S^{2}N will produce a sequence of segments; some will be matched to a ground truth segment and the rest will not be. Ideally, we want the matched segments to have high confidence value, whereas the unmatched ones have low confidence value. The loss function for confidence estimation satisfies this desired criterion. Recall that the S^{2}N predicts a confidence value c_n at each decoding step n, and δ_n indicates whether there is a matching target segment for this decoding step. We use the cross-entropy loss to measure the compatibility between c_n and δ_n:

$$\mathcal{L}_{conf}(S_n, \delta_n) = -\delta_n \log(c_n) - (1 - \delta_n) \log(1 - c_n). \quad (8)$$

4.2 Assignment Strategy: Hungarian Matching with Lexicographic Cost (HMLC)

The last important component of the loss function in Eq. (4) evaluates the matching between the target segments $\mathcal{G}$ and the predicted ones $\mathcal{S}$. In this section, we propose to use Hungarian Matching with Lexicographic Cost (HMLC), a bipartite matching strategy inspired by [16]. Specifically, we define the matching cost between a predicted segment S_n and a ground truth G_k using a triplet cost function:

$$\Delta(G_k, S_n) = (o_{kn}, n, l_{kn}). \quad (9)$$

The function $\Delta : \mathcal{G} \times \mathcal{S} \rightarrow \Re^3$ returns a tuple where l_{kn} is the $L1$ distance between G_k and S_n. n is an integer indicating the output order of S_n (the earlier the output, the lower the cost). o_{kn} is the penalty on whether there is significant overlap between G_k and S_n:

$$o_{kn} = \begin{cases} 0 & \text{if } IoU(G_k, S_n) \geq 0.5 \\ 1 & \text{otherwise.} \end{cases} \quad (10)$$

We use the Hungarian algorithm [55] to determine the best matching according to lexicographic order:

$$\begin{aligned} f^* &= \underset{f}{\text{argmin}} \sum_{k=1}^{K} \Delta(G_k, S_{f(k)}) \\ &= \underset{f}{\text{argmin}} \left(\sum_{k=1}^{K} o_{kf(k)}, \sum_{k=1}^{K} f(k), \sum_{k=1}^{K} l_{kf(k)} \right). \end{aligned} \quad (11)$$

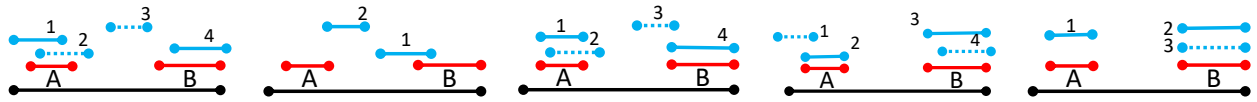
The Hungarian algorithm first selects a set of predictions that significantly overlap ($IOU \geq 0.5$) with the ground truth (i.e., using o). For tie-breaking (*i.e.*, when there exist multiple subsets that significantly overlap with the ground truth), it will first consider the order of segments (i.e., n), and finally the exact amount of overlap (i.e., l) if necessary. We illustrate the behaviour of HMLC in the next section.

4.3 Illustrative Assignment Examples

An appropriate assignment strategy is crucial for S^{2}N to generate meaningful results in the expected order. For many applications, we want the S^{2}N to: (1) generate segments that have high overlap with the ground truth, and (2) generate true positive segments earlier than false positive ones. In the following examples, we show how the proposed assignment strategy satisfies these two criteria.

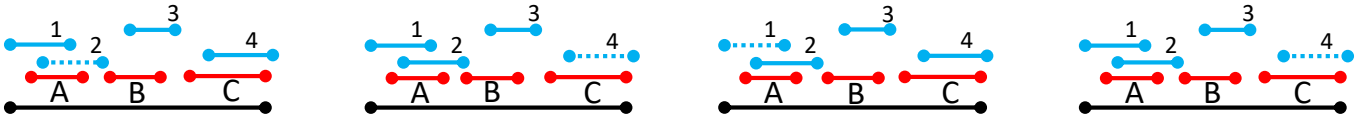
Let us start with the situation shown in Figure (3a), in which the black line at the bottom is an input sequence and the red lines A and B are two ground truth segments. Assume for now S^{2}N outputs sequentially four segments: 1, 2, 3 and 4, in that order. In this example, the IOU scores between the predicted segments 1, 2, 3, 4 and the target segments A and B are shown in Table 1. Based on the scores, matching 1 to A and 4 to B gives the lowest lexicographic cost.

In general, the first two components (o_{kn} and n) of the lexicographic cost triplet (Eq. 9) are sufficient for most matching problems. However, in the early training stage, it is possible that none of the predicted segments has significant overlap with a ground truth segment, as shown in Figure 3b. In this case, the third component of the triplet cost function



(a) (b) (c) (d) (e)

Fig. 3: Illustration of the HMLC assignment strategy. Given an input frame sequence (black line), the segments in red (A, B) represent the target segments and the segments in blue (1, 2, ...) are the sequentially generated segments. The solid blue lines indicate segments that have been matched with ground truth whereas the dashed ones indicate false positives. (a)-(e): five examples of HMLC matching of generated segments to ground truth. HMLC encourages S^{2}N to generate segments that are well aligned with the ground truth and output true positive segments earlier than false positive ones. (a, b), initial matches, (c, d) final non-optimal matches (to be eliminated by non-maximum suppression), (e) optimal match, segments are well aligned with ground truth and true positives appear earlier than false positives.



(a) HMLC (b) Fix (c) Greedy (d) TopK (3)

Fig. 4: Illustration of alternative assignment strategies. The notations are the same as Fig. 3. Different assignment strategies will assign the predicted segments to the ground truth segments differently; solid blue lines indicate matched predictions, dashed blue lines indicate unmatched predictions. (a) HMLC matching (1-A, 3-B, 4-C); (b) Fixed order matching (1-A, 2-B, 3-C); (3) Greedy matching (2-A, 3-B, 4-C); (4) TopK matching for $K = 3$ (2-A, 3-B, 1-C)

	Predicted segments			
	1	2	3	4
A	(0, 1, 0.3)	(0, 2, 0.2)	(1, 3, 1.0)	(1, 4, 1.0)
B	(1, 1, 1.0)	(1, 2, 1.0)	(1, 3, 1.0)	(0, 4, 0.2)

TABLE 1: Lexicographic Cost between generated segments and ground truth in Figure 3a

is used to encourage small and stable adjustments of the predicted segments toward the targets.

Note that matching with HMLC does not guarantee that the loss function optimization results in the expected output. Theoretically it is possible to have the trained S^{2}N output results like the ones in Figure 3c or Figure 3d, which are not optimal in the ordering of the output segments. However, by designing the matching algorithm to prefer early predictions, these situations were rare: in the initial training stages the cost induced by the order of the segments plays a more important role in determining the matching. Additionally, such non-optimal situations can be easily discarded by a simple non-maximum suppression in post-processing.

Similarly, for the score predictor, since early outputs are likely to be matched with ground truth, the expected scores of the early predictions are more likely to be close to 1.

4.4 Alternative Assignment Strategies

HMLC matching is rather complex but it was found to be very important in our experiments. We discuss simpler alternatives and their limitations.

Fixed Order Matching. Perhaps the most straightforward matching strategy is to match the output of S^{2}N sequentially to the target segments following a natural ordering, *e.g.*, from left to right. We label this matching *FIX*. For example, for all the cases shown in Fig. 3, fixed matching simply assigns 1 to A and 2 to B regardless of their positions. The limitation of fixed order matching is that it assigns candidate hypotheses to ground-truth segments ignoring their positions. One such sub-optimal example is shown in Figure 4b.

Greedy Matching: Another strategy is to simply match the generated segments to the closest targets solely based on their distance, by zeroing out the first and second terms in the HMLC triplet. We label this strategy as $Greedy$. The limitation of greedy matching is that it does not enforce true positive segments to be output early, as shown in Figure 4c.

Top K Matching Top K matching (labeled as *TopK*) is a simplified version of HMLC where the first item in the triplet cost is zeroed out, *i.e.*, the early outputs are always matched to ground truth segments and the late outputs are always ignored, as shown in Figure 4d. This reduces the flexibility of S^{2}N to predict the optimal output.

We will compare empirical matching results in Sec 5.5.

5 Experiments

In this section we show that S^{2}N achieves state of the art performance on three different tasks: video summarization, video highlighting, and action proposal generation. Detailed analysis will be provided for all cases, with a slight emphasis on the task of action proposal generation as the more complex of the three.

5.1 Model Implementation and Hyper-parameters

We used the same architecture in all experiments even though we expect even better results can be achieved by tuning the model for each problem. Unless otherwise specified, the

encoder is a two layer bi-directional GRU with 512 hidden units with dropout rate 0.5, the GRU module in SDU is one-directional with 1024 hidden units. All the models are trained with the Adam optimizer [56] for 50 epochs with an initial learning rate of 0.0001, which was decreased by a factor of 10 when the training performance plateaued, batch size of 32 and L_2 gradient clipping of 1.0. The trade-off factor α in Eq. (4) is set specifically for different applications to ensure that $\mathcal{L}_{loc}$ does not dominate the total loss.

A weight adjustment for the score predictor is also used if necessary to account for the imbalance between the positive and negative samples.

Note that S^{2}N is not restricted to the input length, essentially it can deal with segmentations with arbitary length, however, due to difficulties of LSTM/GRU keeping track of long sequences and simplicity of processing, we split each input video into overlapping chunks of the same length.

5.2 Video Highlight Detection

Video highlight detection aims to detect one temporal clip which is the most salient event for each video. In this section, we show that S^{2}N can be trained to be a highlight detector by generating one segment per video.

Dataset. We adopted the large-scale VTW dataset [27] for video highlight detection. The VTW dataset contains 18100 videos, originally proposed for a video captioning task. A subset of the dataset was labeled with temporal locations of highlight shots. We followed the same train/test split as in [6]: the first 1500 videos were used for training, and the next 500 videos for testing. The average video duration is 1.5 minutes, approximately 2000 frames for each video.

Implementation. For each video, we extracted I3D features [57] on RGB frames. Following [39], we rescale the feature sequence of each video to new length of 150 by linear interpolation. Since each video only contains one highlight segment, we set the number of segments to 1 for S^{2}N.

Evaluation metric. We followed the commonly used F1-scores to evaluate the similarity between the predicted highlight segments and ground truth segments.

Baselines. We compared S^{2}N with three state-of-the-art video highlight detection algorithms on the VTW dataset: DPP-LSTM [4], Hierarchical-RNN [30], and retrospective sequence-to-sequence model [6]. Both [30] and [6] adopt a hierarchical model to extract features from videos: first the original video is divided into multiple segments, within each segment, an RNN-based encoder is used to extract features for the segment; then a higher level RNN is used to incorporate segment-wise features. However, these methods require additional parameters and increase the probability of overfitting. In our method, we forego the hierarchical structures used in previous models to combine local and global features. Instead, for simplicity, we directly input to our model the features extracted from pre-trained networks [57].

Results. Table 3 shows the F1 scores of the proposed S^{2}N and other methods. As can be seen, S^{2}N achieves the best performance.

Ablation Studies. To understand the importance of the squared EMD loss, we performed controlled experiments by replacing the squared EMD loss in S^{2}N with other alternatives. We compare with the cross entropy loss and the L_2 loss discussed in 4.1. Clearly, the proposed squared EMD loss is able to guide the pointer network to point to the correct start and end positions.

We have also experimented with the original pointing mechanism proposed in [15]. Specifically, rather than using Eq. 1 that localizes the boundaries based on the correlation between hidden states and decoding states, the implementation in [15] is based on the input features: $b_n = \text{argmax}_i\, g(\mathbf{h}_n, \mathbf{x}_i)$. We achieved significantly worse results following [15]. This suggests that compared to the input feature $\mathbf{x}_i$, the hidden state $\mathbf{h}_i$ captures more progressive information, which is crucial for S^{2}N.

TABLE 2: F1 scores (%) of various video highlight detection methods on the VTW dataset

DPP-LSTM [4]	H-RNN [30]	re-seq2seq [6]	S^{2}N (proposed)
44.3	46.9	48.0	**48.8**

TABLE 3: F1 scores (%) of S^{2}N when different loss functions are used

Cross Entropy Loss	L2 Loss	squared EMD Loss (proposed)
41.5	46.5	**48.8**

5.3 Video Summarization

Automatic video summarization provides a method for humans to browse and analyze video data. Different than video highlight detection algorithms that only output one segment per video, a video summarization algorithm needs to select a small set of segments that are interesting, diverse, and representative of the original video. In this section, we show that S^{2}N can be trained to summarize long videos by generating a set of segments.

Dataset. We performed experiments on SumMe [58], a standard benchmark for video summarization. SumMe consists of 25 user videos covering various topics such as holidays and sports. Each video in SumMe ranges from 1 to 6 minutes and is annotated by 15 to 18 people (thus there are multiple ground truth summaries for each video). We treated each annotation separately and considered all of them ground truth. In this way, S^{2}N was trained to model multiple segment combinations to account for different user annotations (around 450 annotated video instances). We used the canonical setting suggested in [4] for evaluation: we used the standard 5-fold cross validation (5FCV), i.e., 80% of videos are for training and the rest for testing.

Implementation. Similar to the video highlight detection task, instead of using a hierarchical model to extract segment-wise features, we directly extract I3D features [57] on RGB frames without fine-tuning. Each video is split into overlapping chunks of 800 frames, subsampled every 8 frames as inputs. We limit the maximum number of output segments to 8.

To generate a summary, we followed standard practice [4], [5] and selected segments based on their scores by maximizing the total scores while ensuring that the length of the

summary does not exceed a limit (typically 15% of the video duration [59]). The maximization step is equivalent to the 0/1 Knapsack problem. Due to the limited amount of training data in SumMe, we trained each split for exactly 10 epochs. We report performance based on the last epoch.

Evaluation metric. We followed the commonly used protocol from [4], [5], [60]: we computed the F1 score to assess the similarity between the predicted segments and the ground truth summaries. To deal with the existence of multiple ground truth summaries [60], we evaluated the predictions with respect to the nearest human summary, i.e., the one that was the most similar to the automatically created one.

Baselines. We compared S^2N to multiple state-of-the-art video summary algorithms including interestingness-based summary [58], submodularity-based summary [60], and recent deep learning based models: DPP-LSTM [4] (based on LSTM and a determinantal point process [61]), GAN_{sup} [31] (based on a GAN [62] and an extra supervision branch), and DR-DSN_{sup} [5] (based on reinforcement learning).

Results. As shown in Table 4, S^2N outperforms all other methods. S^2N is designed to capture all the information needed for generating good summaries. We also visualize a summarization example in Figure 5. Apart from the experiments with I3D features, we have tested C3D features [63] under the same setting and got an F1 score of 43.1, which is again superior to baseline scores, showing that S^2N is quite robust to the choice of features.

In our experiments, we observed that our algorithm was not sensitive to the parameter that specified the number of segments, as can be seen in Table 5. As a whole, Tables 4 and 5 demonstrate that S^2N outperforms the previous state-of-the-art models for all parameter settings.

5.4 Temporal Action Proposal

Temporal Action Proposal (TAP) generation, akin to object proposal generation for images, is an important problem as accurate extraction of semantically important segments (e.g., human actions) from untrimmed videos is an important step for large-scale video analysis. In contrast to the previous applications, TAP has a slightly different goal; to generate a specific amount of high quality proposals that cover the action events with both high recall and high precision (based on temporal overlap). In this section we show that an S^2N can be trained to generate action proposals.

Dataset. We evaluated S^2Ns on the THUMOS14 dataset [64], a challenging action proposal benchmark. Following standard practice, we trained an S^2N on the validation set and evaluated it on the testing set. On these two sets, 200 and 212 videos respectively have temporal annotations across 20 classes. The average video duration in THUMOS14 is 233 seconds. The average number of labeled actions in each video is around 15, making the task particularly challenging. The average action duration is 4 seconds and more than 99% of the actions are under 10 seconds. We trained an S^2N using 180 out of 200 videos from the validation set, after randomly selecting 20 videos for validation.

Metrics. We evaluated S^2N under AR-N (our primary metric, as it was widely used in multiple previous works) . Following [37], we used additional metrics to further investigate the performance of S^2N. In detail:

- *AR-N* [65], [39], [36], [66]: this measures the average recall (AR) as a function of the number of proposals per video. Note that we retrieve the same number of proposals (N) for all the test videos, regardless of video length.
- *AR-F* [37]: this measures the average recall (AR) as a function of proposal frequency (F), which denotes the number of retrieved proposals per second for a video. For a video that is L_i seconds long and has proposal frequency F, we retrieve $N_i = F \times L_i$ proposals.
- *Recall@F-tIoU* [37]: this metric measures the recall rate at proposal frequency F across to different temporal Intersections over Union (tIoUs). In the evaluation, we set $F = 1.0$ following [37].
- *Recall@N-tIoU* [37]: this metric measures the recall rate when N proposals are retrieved, across different tIoUs.

Features. We used C3D features [63] to fairly compare with previous methods. Following [45], [36], we used activations from the top layer of a 3D convolutional network trained for action classification, then we performed PCA dimensionality reduction to improve computational performance.

Implementation. We split each video into overlapping chunks of 360 frames (~12s) and subsampled every 4 frames. We set the number of proposals generated from each chunk to be 15, as this was the largest possible number of ground truth proposals contained in a chunk during training.

During inference, we combined the proposals from all chunks, sorted them by their scores, and applied Non-Maximum Suppression (NMS). NMS was the only post-processing step used to address the overlap introduced by splitting the videos.

Baselines. We first compare S^2N to state-of-the art TAP generation methods including i) DAPs [36], which use an encoder LSTM and a regression branch for localization, ii) Sparse-prop [35] that applies dictionary learning for class independent proposal generation over a large set of candidate proposals, iii) TURN-TAP [37] that evaluates candidate proposals in a sliding window manner over different temporal scales and level of contexts (we compare with variants of TURN-TAP based on different features and denote them as TURN-C3D and TURN-FLOW). We also compare with *sliding window* and *random* generators. For the DAPs, Sparse-prop, and TURN-TAPS, we plot the curves using the generated proposals provided by the authors. Both sliding window and random proposals are generated following Gao *et al*. [37].

Results. The performance of various methods under different metrics is shown in Figure 6. S^2N outperforms the other methods by a significant margin on all metrics. The performance gap between S^2N and DAPs might imply the usefulness of contextual information.

Error Analysis To further investigate S^2N, we performed quantitative analysis [67] to understand the types of errors often made by S^2N and how much each specific error type affected the performance of the algorithm.

TABLE 4: $F1$ scores (%) of various video summary methods on the SumMe dataset [58]

Interestingness [58]	Submodularity [60]	DPP-LSTM [4]	GAN $_{sup}$ [31]	DR-DSN $_{sup}$ [5]	S^{2}N (proposed)
39.4	39.7	38.6	41.7	42.1	**44.6**

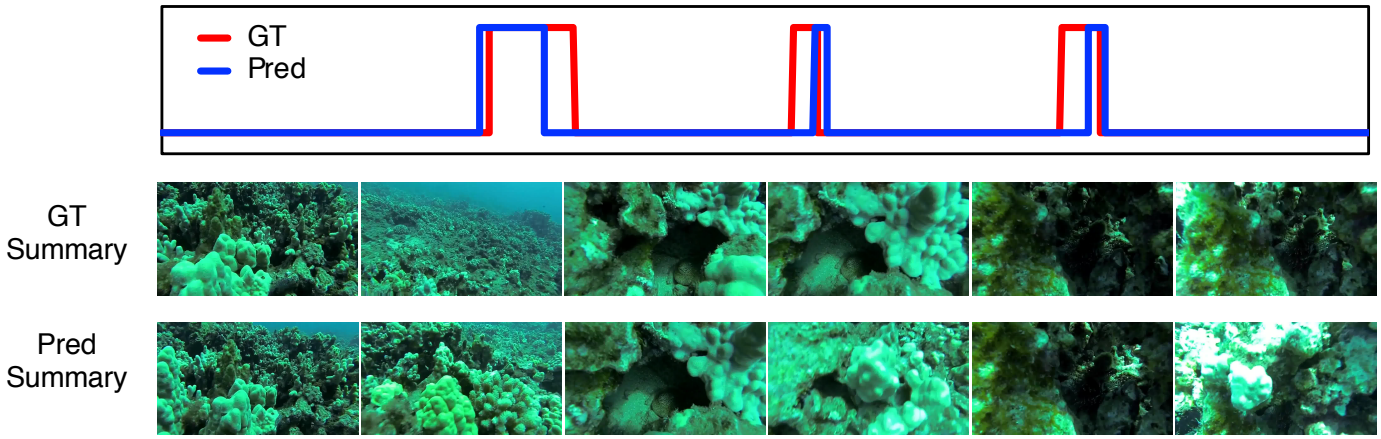
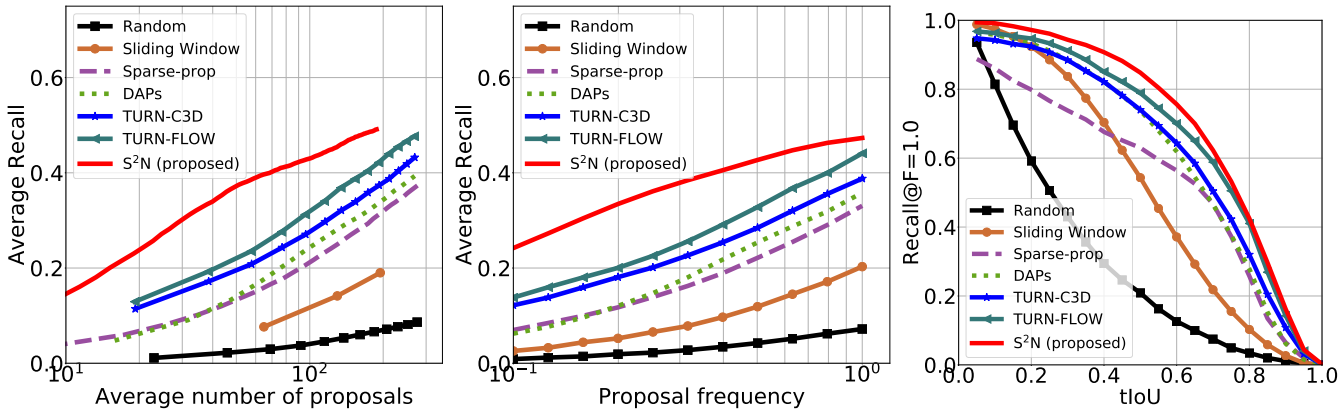


Fig. 5: Visualization of the summarization results. S^{2}N localizes the interesting events in the video, as previously annotated.



(a) AR-N (b) AR-F (c) Recall@1.0-tIoU

Fig. 6: S2N with C3D features outperforms previous temporal action proposal generation approaches on THUMOS-14 under various performance metrics.

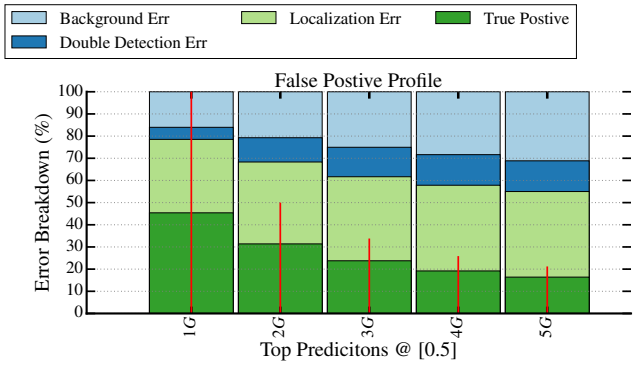
TABLE 5: F1 scores of S^{2}N for different numbers of output segments on SumMe [58] dataset. S^{2}N is not very sensitive to this hyper parameter

#Outputs	5	6	7	8	9	10
F1	44.3	44.2	43.9	44.6	41.8	42.5

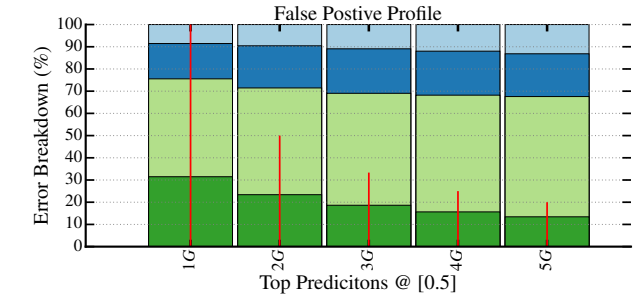
Specifically, we followed the notions in Sec. 4.1. A proposal S_n was a **True Positive** if and only if $\exists G_m \in \mathcal{G}$ such that S_n was the highest scoring prediction with $tIoU(G_m, S_n)$ larger than a threshold. Otherwise, S_n was a **False Positive** and we classified it into the following three categories: (1). *Double Detection Error*: Two distinct predictions satisfied the tIoU threshold for the same ground truth instance (higher scoring one is selected); (2). *Localization Error*: A prediction that had a minimum 0.1 tIoU but failed to meet the tIoU threshold with the ground truth instance; (3). *Background Error*: A prediction that did not meet a minimum 0.1 tIoU with any ground truth instance.

We executed our analysis on the error profile of the top-5G predictions, where G was the number of ground truth instances because 5G was already large enough for S^{2}N proposals to cover most of the positives. We compared S^{2}N with C3D features to TURN-C3D [37] (one of the best competing methods) with results shown in Figure 7. In addition to a higher positive rate, we observe the following: (1) even with a very limited number of proposals ($\leq 5G$), S^{2}N covered the majority of ground truth segments (around 90%); (2) the relatively smaller portion of double detection error showed that S^{2}N tended to produce more diverse results than its alternatives.

Speed. S^{2}N is efficient since it does not require repeated computation over multi-scale context. Specifically, S^{2}N processes each frame in a sequence only once in the encoding stage and sequentially outputs multiple segments over the whole sequence in the decoding stage. It is more efficient than recent models ([45], [66]) that evaluate on a dense set



(a) Error Analysis of S^{2}N with C3D features



(b) Error Analysis of TURN-C3D

Fig. 7: Comparing different action proposal methods. The red vertical lines indicate the number of true positives (100% for 1G, 50% for 2G ...). Best viewed on a screen.

of highly-overlapped candidates at each temporal step in a sequence. Quantitatively, it takes on average 0.028s to process a 12s, 30FPS video on a GTX Titan X Maxwell GPU with 12GB memory. In batch mode, it takes around 2s to generate over 1200 proposals for an 8-minute video (14400 frames sampled every 4 frames). This is more than two times faster than recently proposed models (1800 FPS *v.s.* 701 FPS [66], 308 FPS [45], 134 FPS [36]).

5.5 Improving S^{2}N for Temporal Action Proposal Generation

In this section, we will describe several techniques to further advance the performance of S^{2}N. In particular, we will replace C3D features by the more recent I3D features [57], and introduce beam search to extend the recall range of the method.

Using more sophisticated features. So far, we have used S^{2}Ns with C3D features, so the resulting S^{2}N models are directly comparable to other existing methods that also used C3D features. As we have shown in the previous section, S^{2}Ns with C3D features have improved the state-of-the-art. However, the underlying C3D features are no longer the state-of-the-art for human action recognition; there are more recent and advanced features such as I3D [57]. In this section, we evaluate S^{2}N with I3D, aiming to further improve state-of-the-art performance. I3D [57] is a two-stream model that is built upon image classification architectures [70] where filters were inflated from 2D to 3D, leading to very deep, naturally spatio-temporal models. We used the model of [57], pre-trained on the Kinetics dataset [71] for action classification. This model inputs a stack of 64 RGB/optical flow frames and extracts a 1024-dimensional feature vector. We extracted both RGB and optical flow features for each frame. We then performed PCA to reduce dimensionality. The input to S^{2}N is the concatenation of two 500-dimensional RGB/optical flow feature vectors.

Extending the range of recall values with beam search. S^{2}N usually decodes a limited number of proposals for each input sequence (generally ≤ 300 segments for a video). This is fine for tasks such as video highlighting or video summarization. However, it makes S^{2}N less competitive for temporal action proposal tasks in which more proposals are allowed.

To address the problem of generating insufficient proposals, we propose a simple but effective modification during inference, that uses *Beam Search*: for each decoding step, instead of only generating one proposal S_n from the maximum response locations(Eq. 1), we make S^{2}N generate N proposals from the top N maximum response locations. The score of a segment $S_n = (b_n, d_n, c_n)$, which previously was only determined by the output of the score predictor branch (Sec. 3.3) is now a product of the score predictor and the response scores for the starting and ending position, *i.e.*:

$$\bar{c}_n = c_n \cdot \sigma(g(\mathbf{h}_n, \mathbf{e}_{b_n})) \cdot \sigma(g(\mathbf{h}_n, \mathbf{e}_{d_n})) \tag{12}$$

where $g(\cdot,\cdot)$ is defined in Eq. 2 and $\sigma(\cdot)$ is the softmax function.

In this way, this *S2N-Beam* can generate N times more proposals than the original S^{2}N. We set N to 10 for all experiments.

Results. Table 6 shows a complete comparison between variants of S^{2}N and current state-of-the-art algorithms under the AR-N metric. In addition to the set of baselines described in Sec. 5.4, we also include Boundary-Sensitive Network (BSN) [39], a recently published multiple-stage pipeline that detects segment boundaries and groups them as proposals. As shown in Table 6, using the same features, the S^{2}N methods significantly outperform the others. Notably, on average S^{2}N-Beam with I3D features performs 5% better than BSN. Another interesting observation is that when N is small in AR-N metric (N=50), using beam search may hurt performance due to multiple candidates proposed.

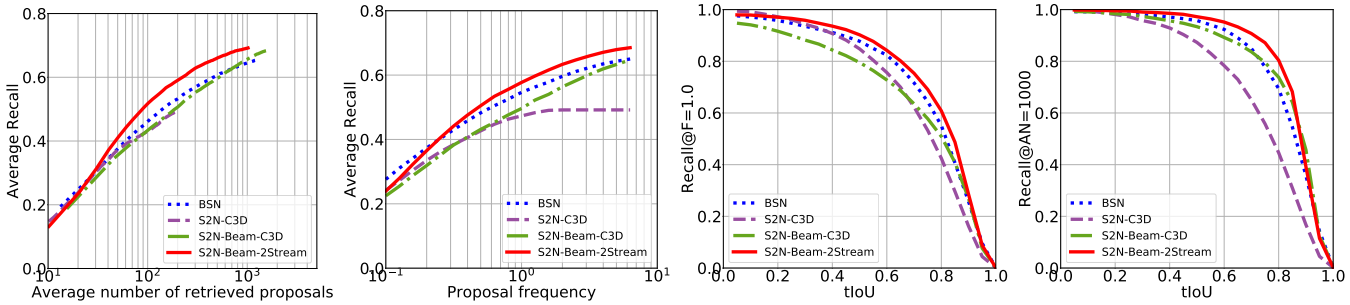
Figure 8 plots the complete performance curves of several methods: BSN, S2N-C3D, S2N-Beam-C3D, and S2N-Beam-I3D. In addition to the performance curves under the AR-N metric (Figure 8a), this figure also shows the performance curves under other metrics: AR-F, Recall@F-1.0 and Recall@N=1000. As shown in the figure, using beam search (S^{2}N-Beam) improves performance especially when the proposal frequency is high.

5.6 Ablation Study on Assignment Strategies

We further explore the influence of different label assignment strategies on the performance of S^{2}Ns. Specifically we compare the proposed S^{2}N trained with the HMLC matching strategy with the alternatives introduced in Section 4.4. As shown in Table 7, the S^{2}N model trained with HMLC consistently outperforms its variants for all settings output proposal numbers.

TABLE 6: Comparison between S^{2}N and the other state-of-the-art proposal generation methods on THUMOS14 in terms of Average Recall at N (AR-N). Methods based on I3D features outperform the ones based on C3D features. Using Beam Search also improves overall performance.

Feature	Method	@50	@100	@200	@500	@1000
-	Random	2.47	4.44	7.42	13.09	16.11
C3D	DAPs [36]	13.56	23.83	33.96	49.29	57.64
C3D	SCNN-prop [35]	17.22	26.17	37.01	51.57	58.20
C3D	TURN [37]	19.63	27.96	38.34	53.52	60.75
C3D	BSN + Greedy-NMS [39]	27.19	35.38	43.61	53.77	59.50
C3D	BSN + Soft-NMS [39]	29.58	37.38	45.55	54.67	59.48
C3D	S^{2}N	**37.06**	43.04	49.18	-	-
C3D	S^{2}N-Beam	36.80	**44.23**	**52.11**	**59.59**	**64.46**
Flow	TURN [37]	21.86	31.89	43.02	57.63	64.17
Two-stream [68]	TAG [38]	18.55	29.00	39.61	-	-
Two-stream [69]	BSN + Greedy-NMS [39]	35.41	43.55	52.23	61.35	65.10
Two-stream [69]	BSN + Soft-NMS [39]	37.46	46.06	53.21	60.64	64.52
I3D [57]	S^{2}N-Beam	**41.29**	**51.96**	**59.19**	**66.01**	**69.11**



(a) AR-N (b) AR-F (c) Recall@1.0-tIoU (d) Recal@N=1000

Fig. 8: Comparison between variants of S^{2}N on THUMOS14. S^{2}N-Beam improve performance especially when proposal frequency is high. S^{2}N-Beam with two stream features achieves best performance under various performance metrics.

Method	@50	@100	@200	@500	@1000
Fix	31.44	41.75	49.23	57.82	61.92
Greedy	34.10	41.82	49.71	58.38	64.03
TopK	35.90	42.98	51.52	58.62	62.55
HMLC	**36.80**	**44.23**	**52.11**	**59.59**	**64.46**

TABLE 7: Comparison of S^{2}N trained with different assignment strategies under metric AR-N. S^{2}N with with HMLC strategy outperforms the alternatives over different number of predefined proposals.

6 Conclusions and Future Work

We have proposed Sequence-to-Segments Network (S^{2}N), a novel architecture that uses Segment Detection Units (SDU) to detect segments sequentially from an input sequence. We have shown that S^{2}N can be applied to multiple real-world problems and achieve state-of-the-art performance.

There are a few directions for future work. One direction is to augment the encoding stage to be capable of recording longer sequences [72]. Another possible direction is to extend S^{2}N to more complex problems such as action detection in untrimmed videos. A third direction is to introduce auxiliary losses to enforce explicit semantic constraints on S^{2}N [66]. It is also possible to base S^{2}N on the fully convolutional encoder-decoder architecture [73], [74].

Acknowledgments

This project was partially supported by NSF-CNS-1718014, NSF-IIS-1763981, NSF-IIS-1566248, the Partner University Fund, the SUNY2020 Infrastructure Transportation Security Center, and a gift from Adobe.

References

[1] M. H. Nguyen, T. Simon, F. De la Torre, and J. Cohn, "Action unit detection with segment-based SVMs," in *Proc. CVPR*, 2010.
[2] M. Hoai, Z.-Z. Lan, and F. De la Torre, "Joint segmentation and classification of human actions in video," in *Proc. CVPR*, 2011.
[3] M. Hoai and F. De la Torre, "Max-margin early event detectors," *IJCV*, vol. 107, no. 2, pp. 191–202, 2014.
[4] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman, "Video summarization with long short-term memory," in *Proc. ECCV*, 2016.
[5] K. Zhou and Y. Qiao, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in *Proc. AAAI*, 2017.
[6] K. Zhang, K. Grauman, and F. Sha, "Retrospective encoders for video summarization," in *Proc. ECCV*, 2018.
[7] H. Yang, B. Wang, S. Lin, D. Wipf, M. Guo, and B. Guo, "Unsupervised extraction of video highlights via robust recurrent auto-encoders," in *Proc. ICCV*, 2015.
[8] M. Sun, A. Farhadi, and S. Seitz, "Ranking domain-specific highlights by analyzing edited videos," in *Proc. ECCV*, 2014.
[9] M. H. Nguyen, L. Torresani, F. De la Torre, and C. Rother, "Weakly supervised discriminative localization and classification: a joint learning process," in *Proc. ICCV*, 2009.
[10] M. Hoai, L. Torresani, F. De la Torre, and C. Rother, "Learning discriminative localization from weakly labeled data," *Pattern Recognition*, vol. 47, no. 3, pp. 1523–1534, 2014.

[11] M. Hoai and F. De la Torre, "Structured prediction for event detection," in *Advanced Structured Prediction*, S. Nowozin, P. Gehler, J. Jancsary, and C. Lampert, Eds. MIT Press, Cambridge, MA, USA, 2014.

[12] N. Peng and M. Dredze, "Named entity recognition for chinese social media with jointly trained embeddings," in *Proc. EMNLP*, 2015.

[13] D. R. Kelley, Y. A. Reshef, D. Belanger, C. McLean, J. Snoek, and M. Bileschi, "Sequential regulatory activity prediction across chromosomes with convolutional neural networks," *Genome research*, 2018.

[14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NIPS*, 2014.

[15] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *NIPS*, 2015.

[16] R. Stewart, M. Andriluka, and A. Y. Ng, "End-to-end people detection in crowded scenes," in *Proc. CVPR*, 2016.

[17] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks," in *Proc. ICML*, 2006.

[18] D. Rumelhart, G. Hinton, and R. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing*. MIT Press, Cambridge, MA, 1986, vol. 1, ch. 8, pp. 318–362.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2014.

[20] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv:1410.5401*, 2014.

[21] O. Vinyals, **L**. Kaiser, T. Koo, S. Petrov, I. Sutskever, and G. Hinton, "Grammar as a foreign language," in *NIPS*, 2015.

[22] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. CVPR*, 2015.

[23] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. CVPR*, 2015.

[24] D. Yow, B.-L. Yeo, M. Yeung, and B. Liu, "Analysis and presentation of soccer highlights from digital video," in *Proc. ACCV*, 1995.

[25] H. Tang, V. Kwatra, M. E. Sargin, and U. Gargi, "Detecting highlights in sports videos: Cricket as a test case," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2011.

[26] S. Nepal, U. Srinivasan, and G. Reynolds, "Automatic detection of'goal'segments in basketball videos," in *Proc. ACMM*, 2001.

[27] K.-H. Zeng, T.-H. Chen, J. C. Niebles, and M. Sun, "Generation for user generated videos," in *Proc. ECCV*, 2016.

[28] Y. Song, J. Vallmitjana, A. Stent, and A. Jaimes, "Tvsum: Summarizing web videos using titles," in *Proc. CVPR*, 2015.

[29] Z. Ji, K. Xiong, Y. Pang, and X. Li, "Video summarization with attention-based encoder-decoder networks," *arXiv preprint arXiv:1708.09545*, 2017.

[30] B. Zhao, X. Li, and X. Lu, "Hierarchical recurrent neural network for video summarization," in *Proc. ACMM*, 2017.

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

[32] H.-W. Kang, Y. Matsushita, X. Tang, and X.-Q. Chen, "Space-time video montage," in *Proc. CVPR*, 2006.

[33] W.-S. Chu, Y. Song, and A. Jaimes, "Video co-summarization: Video summarization by visual co-occurrence," in *Proc. CVPR*, 2015.

[34] Y. J. Lee, J. Ghosh, and K. Grauman, "Discovering important people and objects for egocentric video summarization," in *Proc. CVPR*, 2012.

[35] F. Caba Heilbron, J. Carlos Niebles, and B. Ghanem, "Fast temporal activity proposals for efficient detection of human actions in untrimmed videos," in *Proc. CVPR*, 2016.

[36] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *Proc. ECCV*, 2016.

[37] J. Gao, Z. Yang, K. Chen, C. Sun, and R. Nevatia, "Turn tap: Temporal unit regression network for temporal action proposals," in *Proc. ICCV*, 2017.

[38] Y. Zhao, Y. Xiong, L. Wang, Z. Wu, X. Tang, and D. Lin, "Temporal action detection with structured segment networks," in *Proc. ICCV*, 2017.

[39] T. Lin, X. Zhao, H. Su, C. Wang, and M. Yang, "Bsn: Boundary sensitive network for temporal action proposal generation," in *Proc. ECCV*, 2018.

[40] Y. Adi, J. Keshet, E. Cibelli, and M. Goldrick, "Sequence segmentation using joint rnn and structured prediction models," in *ICASSP*, 2017.

[41] J. Namikawa and J. Tani, "A model for learning to segment temporal sequences, utilizing a mixture of rnn experts together with adaptive variance," *Neural Networks*, vol. 21, no. 10, pp. 1466–1475, 2008.

[42] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," in *Proc. ICLR*, 2016.

[43] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," in *Proc. EMNLP*, 2014.

[44] K. Yao, T. Cohn, K. Vylomova, K. Duh, and C. Dyer, "Depth-gated recurrent neural networks. arxiv preprint," *arXiv preprint arXiv:1508.03790*, vol. 9, 2015.

[45] S. Buch, V. Escorcia, C. Shen, B. Ghanem, and J. C. Niebles, "Sst: Single-stream temporal action proposals," in *Proc. CVPR*, 2017.

[46] R. Jozefowicz, W. Zaremba, and I. Sutskever, "An empirical exploration of recurrent network architectures," in *Proc. ICML*, 2015.

[47] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv:1412.3555*, 2014.

[48] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *Proc. CVPR*, 2016.

[49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *Proc. ECCV*, 2016.

[50] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016.

[51] N. Srivastava, E. Mansimov, and R. Salakhudinov, "Unsupervised learning of video representations using lstms," in *Proc. ICML*, 2015.

[52] L. Hou, C.-P. Yu, and D. Samaras, "Squared earth mover's distance-based loss for training deep neural networks," *arXiv:1611.05916*, 2016.

[53] S. Shalev-Shwartz and A. Tewari, "Stochastic methods for l1-regularized loss minimization," *Journal of Machine Learning Research*, vol. 12, no. Jun, pp. 1865–1892, 2011.

[54] D. G. Luenberger, *Introduction to linear and nonlinear programming*. Addison-Wesley publishing company, 1973.

[55] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[56] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[57] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *Proc. CVPR*, 2017.

[58] M. Gygli, H. Grabner, H. Riemenschneider, and L. Van Gool, "Creating summaries from user videos," in *Proc. ECCV*, 2014.

[59] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-down neural attention by excitation backprop," in *Proc. ECCV*, 2016.

[60] M. Gygli, H. Grabner, and L. Van Gool, "Video summarization by learning submodular mixtures of objectives," in *Proc. CVPR*, 2015.

[61] B. Gong, W.-L. Chao, K. Grauman, and F. Sha, "Diverse sequential subset selection for supervised video summarization," in *NIPS*, 2014.

[62] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[63] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *Proc. ICCV*, 2015.

[64] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, "THUMOS challenge: Action recognition with a large number of classes," http://crcv.ucf.edu/THUMOS14/, 2014.

[65] G. Yu and J. Yuan, "Fast action proposals for human action detection and search," in *Proc. CVPR*, 2015.

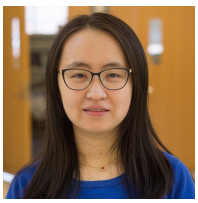
[66] S. Buch, V. Escorcia, B. Ghanem, L. Fei-Fei, and J. Niebles, "End-to-end, single-stream temporal action detection in untrimmed videos," in *Proc. BMVC.*, 2017.

[67] H. Alwassel, F. Caba Heilbron, V. Escorcia, and B. Ghanem, "Diagnosing error in temporal action detectors," in *Proc. ECCV*, 2018.

[68] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *NIPS*, 2014.

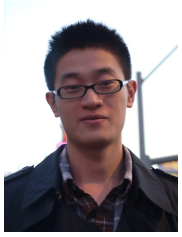
[69] Y. Xiong, L. Wang, Z. Wang, B. Zhang, H. Song, W. Li, D. Lin, Y. Qiao, L. Van Gool, and X. Tang, "Cuhk & ethz & siat submission to activitynet challenge 2016," *arXiv preprint arXiv:1608.00797*, 2016.
[70] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. CVPR*, 2015.
[71] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev *et al.*, "The kinetics human action video dataset," *arXiv preprint arXiv:1705.06950*, 2017.
[72] S. Li, W. Li, C. Cook, C. Zhu, and Y. Gao, "Independently recurrent neural network (indrnn): Building a longer and deeper rnn," in *Proc. CVPR*, 2018.
[73] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *Proc. ICML*, 2017.
[74] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, **L**. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

Zijun Wei received a Bachelor of Network Engineering from Nanjing University of Science and Technology in 2011 and a Master of Robotics from Carnegie Mellon University in 2013. He is currently a PhD candidate at the Department of Computer Science, Stony Brook University. His research focuses on visual saliency and human preference modeling.



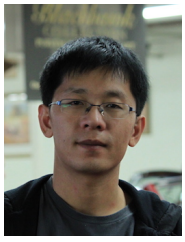
Boyu Wang received a Bachelor of Computer Science from Nanjing University of Posts and Telecommunications in 2014. She is currently a PhD candidate at the Department of Computer Science, Stony Brook University. Her research focuses on human action early recognition and prediction.

Minh Hoai is an Assistant Professor of Computer Science at Stony Brook University and a Principal Research Scientist at VinAI Research. He received Bachelor of Software Engineering from the University of New South Wales in 2005 and Ph.D. in Robotics from Carnegie Mellon University in 2012. His research interests are in computer vision and machine learning, especially human action/activity recognition and prediction.



Jianming Zhang is a research scientist at Adobe. He received his BS and MS degrees in mathematics in 2008 and 2011 respectively at Tsinghua University and a PhD degree in computer science in 2016 at Boston University. His research area is computer vision and machine learning with a special focus on deep learning, visual saliency and image editing.

Xiaohui Shen is a Researcher at ByteDance AI Lab. Before that, he was a Senior Research Scientist at Adobe Research. He obtained his PhD degree from the Department of EECS at Northwestern University, and received the MS and BS degrees from the Department of Automation at Tsinghua University, China. His research interests include computer vision and deep learning.



Zhe Lin is a researcher in Creative Intelligence Lab, Adobe Research. He received a Ph.D. degree in Electrical and Computer Engineering from University of Maryland at College Park in May 2009. Prior to that, he obtained a M.S. degree in Electrical Engineering and Computer Science from Korea Advanced Institute of Science and Technology in August 2004, and a B.Eng. degree in automatic control from University of Science and Technology of China. His research interests include computer vision, image and video processing, machine learning, deep learning, artificial intelligence.

Radomír Měch received his MSc at the Charles University in Prague, Czech Republic and PhD at the University of Calgary, Canada. Now he heads the Procedural Imaging Group at Adobe Research. His areas of research are procedural modeling, with a particular focus on interaction with procedural models and casual modeling, rendering, 3D printing, and image evaluation algorithms.

Dimitris Samaras received his Diploma in computer science and engineering from the University of Patras in 1992, his MSc degree from Northeastern University in 1994, and his PhD degree from the University of Pennsylvania in 2001. He is a SUNY Empire Innovation professor of computer science at Stony Brook University. His research interests include human behavior analysis, illumination modeling and estimation for recognition and graphics, and biomedical image analysis. He is Program Chair of CVPR 2022.