Supplementary Material: Video Summarization with Long Short-term Memory

Ke Zhang[1⋆], Wei-Lun Chao[1⋆], Fei Sha[2], and Kristen Grauman[3]

[1]Dept. of Computer Science, U. of Southern California, United States
[2]Dept. of Computer Science, U. of California, Los Angeles, United States
[3]Dept. of Computer Science, U. of Texas at Austin, United States
{zhang.ke, weilunc}@usc.edu, feisha@cs.ucla.edu, grauman@cs.utexas.edu

In this Supplementary Material, we provide details omitted in the main text:

- Section 1: converting between different formats of ground-truth annotations (Section 3.1 in the main text)
- Section 2: details of the datasets (Section 4.1 in the main text)
- Section 3: details of our LSTM-based models, including the learning objective for dppLSTM and the generating process of shot-based summaries for both vsLSTM and dppLSTM (Section 3.4 and 3.5 in the main text)
- Section 4: comparing different network structures for dppLSTM (Section 3.4 in the main text)
- Section 5: Other implementation details
- Section 6: Additional discussions on video summarization

1 Converting between different formats of ground-truth annotations

As mentioned in Section 3.1 of the main text, existing video summarization datasets usually provide the ground-truth annotations in (one of) the following three formats — (a) selected keyframes, (b) interval-based keyshots, and (c) frame-level importance scores. See Table 1 for illustration.

In order to combine multiple datasets to enlarge the training set, or to enable any (supervised) video summarization algorithm to be trained under different ground-truth formats, we introduce a general procedure to convert between different formats. *Note that we perform this procedure to the ground truths only in the training phase.* In the testing phase, we directly compare with the user-generated summaries in their original formats, unless stated otherwise (see Section 2). Also note that certain conversions require *temporal segmentation* to cut a video into disjoint time intervals, where each interval contains frames of similar contents. Since none of the datasets involved in the experiments provides *ground-truth temporal segmentation*, we apply the kernel temporal segmentation (KTS) proposed by Potapov *et al.* [1]. The resulting intervals are around 5 seconds on average.

⋆ Equal contributions

Table 1. Illustration of different formats of ground-truth annotations for video summarization. We take a 6-frame sequence as an example.

Format	Description
(a) keyframes	{frame 2, frame 6} or [0 1 0 0 0 1]
(b) interval-based keyshots	{frames 1–2, frames 5–6} or [1 1 0 0 1 1]
(c) frame-level importance scores	[0.5 0.9 0.1 0.2 0.7 0.8]

1.1 keyframes → keyshots and frame-level scores

To covert keyframes into keyshots, we first *temporally segment* a video into disjoint intervals using KTS [1]. Then if an interval contains at least one keyframe, we view such an interval as a keyshot, and mark all frames of it with score 1; otherwise, 0.

To prevent generating too many keyshots, we rank the candidate intervals (those with at least one keyframe) in the descending order by the number of key frames each interval contains divided by its duration. We then select intervals in order so that the total duration of keyshots is below a certain threshold (e.g., using the knapsack algorithm as in [2]).

1.2 keyshots → keyframes and frame-level scores

Given the selected keyshots, we can randomly pick a frame, or pick the middle frame, of each keyshot to be a keyframe. We also directly mark frames contained in keyshots with score 1. For those frames not covered by any keyshot, we set the corresponding importance scores to be 0.

1.3 frame-level scores → keyframes and keyshots

To convert frame-level importance scores into keyshots, we first perform *temporal segmentation*, as in Section 1.1. We then compute interval-level scores by averaging the scores of frames within each interval. We then rank intervals in the descending order by their scores, and select them in order so that the total duration of keyshots is below a certain threshold (e.g., using the knapsack algorithm as in [2]). We further pick the frame with the highest importance score within each keyshot to be a keyframe.

Table 2 summarizes the conversions described above.

2 Details of the datasets

In this section, we provide more details about the four datasets — **SumMe** [3], **TVSum** [2], **OVP** [4,5], and **Youtube** [5] — involved in the experiments. Note that **OVP** and **Youtube** are only used to augment the training set.

Table 2. Illustration of the converting procedure described in Section 1.1–1.3. We take a 6-frame sequence as an example, and assume that the temporal segmentation gives three intervals, {frames 1–2, frames 3–4, frames 5–6}. The threshold of duration is 5.

Conversion	Description
Section 1.1	(a) [0 1 0 0 0 1] → (b) [1 1 0 0 1 1], (c) [1 1 0 0 1 1]
Section 1.2	(b) [1 1 0 0 1 1] → (a) [0 1 0 0 0 1], (c) [1 1 0 0 1 1]
Section 1.3	(c) [0.5 0.9 0.1 0.2 0.7 0.8] → (b) [1 1 0 0 1 1], (a) [0 1 0 0 0 1]

(a) keyframes (b) interval-based keyshots (c) frame-level importance scores

2.1 Training ground truths

Table 3 lists the training and testing ground truths provided in each dataset. Note that in training, we require a single ground truth for each video, which is directly given in **SumMe** and **TVSum**, but not in **OVP** and **Youtube**. We thus follow [6] to create a single ground-truth set of keyframes from multiple user-annotated ones for each video.

Table 4 summarizes the formats of training ground truths required by our proposed methods (vsLSTM, dppLSTM) and baselines (MLP-Shot, MLP-Frame). We perform the converting procedure described in Section 1 to obtain the required training formats if they are not provided in the dataset. We perform KTS [1] for temporal segmentation for all datasets.

2.2 Testing ground truths for TVSum

TVSum provides for each video multiple sequence of frame-level importance scores annotated by different users. We follow [2] to convert each sequence into a keyshot-based summary for evaluation, which is exactly the one in Section 1.3. We set the threshold to be 15% of the original video length, following [2].

Table 3. Training and testing ground truths provided for each video in the datasets.

Dataset	Training ground truths	Testing ground truths
SumMe	a sequence of frame-level scores	multiple sets of keyshots
TVSum	a sequence of frame-level scores	multiple sequences of frame-level scores†
OVP	multiple sets of keyframes‡	-
Youtube	multiple sets of keyframes‡	-

† following [2], we convert the frame-level scores into keyshots for evaluation.
‡ following [6], we create a single ground-truth set of keyframes for each video.

Table 4. The formats of training ground truths required by vsLSTM, dppLSTM, MLP-Shot, and MLP-Frame.

Method	Training ground truths
MLP-Shot	shot-level importance scores[†]
MLP-Frame	frame-level importance scores
vsLSTM	frame-level importance scores
dppLSTM	keyframes, frame-level importance scores[‡]

[†] The shot-level importance scores are derived as the averages of the corresponding frame-level importance scores. We perform KTS [1] to segment a video into shots (disjoint intervals).
[‡] We pre-train the MLP $f_I(\cdot)$ and the LSTM layers using frame-level scores.

3 Details of our LSTM-based models

In this section, we provide more details about the proposed LSTM-based models for video summarization.

3.1 The learning objective of dppLSTM

As mentioned in Section 3.4 of the main text, we adopt a stage-wise optimization routine to learn dppLSTM — the first stage is based on the prediction error of importance scores; the second stage is based on the maximum likelihood estimation (MLE) specified by DPPs. Denote Z as a ground set of N items (e.g, all frames of a video), and $\boldsymbol{z}^* \subset \mathsf{Z}$ as the target subset (e.g., the subset of keyframes). Given the N $\times$ N kernel matrix $\boldsymbol{L}$, the probability to sample $\boldsymbol{z}^*$ is

$$P(\boldsymbol{z}^* \subset \mathsf{Z}; \boldsymbol{L}) = \frac{\det(\boldsymbol{L}_{\boldsymbol{z}^*})}{\det(\boldsymbol{L} + \boldsymbol{I})}, \tag{1}$$

where $\boldsymbol{L}_{\boldsymbol{z}^*}$ is the principal minor indexed by $\boldsymbol{z}^*$, and $\boldsymbol{I}$ is the N $\times$ N identity matrix.

In dppLSTM, $\boldsymbol{L}$ is parameterized by $\boldsymbol{\theta}$, which includes all parameters in the model. In the second stage, we learn $\boldsymbol{\theta}$ using MLE [7]

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \sum_i \log\{P(\boldsymbol{z}^{(i)*} \subset \mathsf{Z}^{(i)}; \boldsymbol{L}^{(i)}(\boldsymbol{\theta}))\}, \tag{2}$$

where i indexes the target subset, ground set, and $\boldsymbol{L}$ matrix of the i-th video. We optimize $\boldsymbol{\theta}$ with stochastic gradient descent.

3.2 Generating shot-based summaries for vsLSTM and dppLSTM

As mentioned in Section 3.1 and 3.5 of the main text, the outputs of both our proposed models are on the frame level — vsLSTM predicts frame-level importance scores, while dppLSTM selects a subset of keyframes using approximate

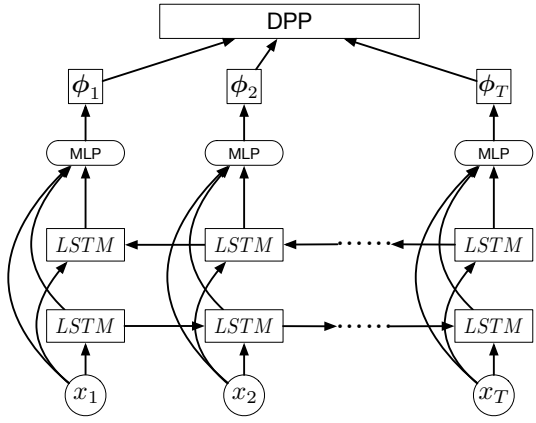

Fig. 1. Our dppLSTM-single model. It is similar to dppLSTM (Fig. 3 in the main text) but learns only a single MLP $f_S(\cdot)$ and then stacks with a DPP.

MAP inference [8]. To compare with the user-annotated keyshots in **SumMe** and **TVSum** for evaluation, we convert the outputs into keyshot-based summaries.

For vsLSTM, we directly apply the conversion in Section 1.3. We set the threshold of the total duration of keyshots to be 15% of the original video length (for both datasets), following the protocols in [2,3,9].

For dppLSTM, we apply the conversion in Section 1.1. In practice, DPP inference usually leads to *high precision* yet *low recall*; i.e., the resulting total duration of keyshots may be far below the threshold (on average, 10%). We thus add in few more keyshots by utilizing the scalar output of the MLP $f_I(\cdot)$, following the procedure in Section 1.3. The MLP $f_I(\cdot)$ is pre-trained using the frame-level importance scores (cf. Section 3.4 of the main text) and conveys a certain notion of importance even after fine-tuning with the DPP objective.

4 Comparing different network structures for dppLSTM

The network structure of dppLSTM (cf. Fig. 3 of the main text) involves two MLPs — the MLP $f_I(\cdot)$ outputting y_t for frame-level importance and the MLP $f_S(\cdot)$ outputting $\boldsymbol{\phi}_t$ for similarity.

In this section, we compare with another LSTM-based model that learns only a single MLP $f_S(\cdot)$ and then stacks with a DPP. We term this model as dppLSTM-single. See Fig. 1 for illustration. dppLSTM-single also outputs a set of keyframes and is likely to generate a keyshot-based summary of an insufficient duration (similar to dppLSTM in Section 3.2). We thus add in few more keyshots by utilizing the diagonal values of $\boldsymbol{L}$ as frame-level scores, following [10].

Table 5 compares the performance of the two network structures, and dppLSTM obviously outperforms dppLSTM-single. As a well-learned DPP model

Table 5. Comparison between dppLSTM and dppLSTM-single on different settings.

Dataset	Method	Canonical	Augmented	Transfer
SumMe	dppLSTM	38.6±0.8	42.9±0.5	41.8±0.5
	dppLSTM-single	37.5±0.9	41.4±0.8	40.3±0.9
TVSum	dppLSTM	54.7±0.7	59.6±0.4	58.7±0.4
	dppLSTM-single	53.9±0.9	57.5±0.7	56.2±0.8

should capture the notions of both quality (importance) and diversity [7], we surmise that separately modeling the two factors would benefit, especially when the model of each factor can be pre-trained (e.g, the MLP $f_I(\cdot)$ in dppLSTM).

5 Other implementation details

In this section, we provide the implementation details for both the proposed models (vsLSTM, dppLSTM) and baselines (MLP-Frame, MLP-Shot).

5.1 Input signal

For vsLSTM, dppLSTM, and MLP-Frame, which all take frame features as inputs, we uniformly subsample the videos to 2 fps[1]. The concatenated feature (of a 5-frame window) to MLP-Frame is thus equivalent to taking a 2-second span into consideration. For MLP-Shot, we perform KTS [1] to segment the video into shots (disjoint intervals), where each shot is around 5 seconds on average.

5.2 Network structures

$f_I(\cdot)$ and $f_S(\cdot)$ are implemented by one-hidden-layer MLPs, while MLP-Shot and MLP-Frame are two-hidden-layer MLPs. For all models, we set the size of each hidden layer of MLPs, the number of hidden units of each unidirectional LSTM, and the output dimension of the MLP $f_S(\cdot)$ all to be 256. We apply the sigmoid activation function to all the hidden units as well as the output layer of MLP-Shot, MLP-Frame, and $f_I(\cdot)$. The output layer of $f_S(\cdot)$ are of linear units. We run for each setting and each testing fold (cf. Section 4.2 of the main text) 5 times and report the average and standard deviation.

5.3 Learning objectives

For MLP-Frame, MLP-Shot, vsLSTM, and the first stage of dppLSTM, we use the square loss. For dppLSTM-single and the second stage of dppLSTM, we use the likelihood (cf. (2)).

[1] For videos with slow varying contents such as SumMe/TVSum, this scheme seems adequate. For OVP and YouTube, even 1 fps is sufficient for fairly good summarization [5,6,10].

5.4 Stopping criteria

For all our models, we stop training after K consecutive epochs with descending summarization F-score on the validation set. We set $K = 5$.

6 Additional discussions on video summarization

Video summarization is essentially a structured prediction problem and heavily relies on how to model/capture the sequential (or temporal) structures underlying videos. In this work, we focus on modeling the structures making sequentially inter-dependent decisions at three levels: (a) realizing boundaries of sub-events/shots; (b) removing redundant nearby shots/frames; (c) retaining temporally distant events despite being visually similar (cf. the motivating example of "leave home" in Section 1 of the main text). Essentially, any decision including or excluding frames is dependent on other decisions made on a temporal line.

References

1. Potapov, D., Douze, M., Harchaoui, Z., Schmid, C.: Category-specific video summarization. In: ECCV. (2014) 1, 2, 3, 4, 6
2. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: Tvsum: Summarizing web videos using titles. In: CVPR. (2015) 2, 3, 5
3. Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: ECCV. (2014) 2, 5
4. : Open video project. http://www.open-video.org/ 2
5. de Avila, S.E.F., Lopes, A.P.B., da Luz, A., de Albuquerque Araújo, A.: Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. Pattern Recognition Letters **32**(1) (2011) 56–68 2, 6
6. Gong, B., Chao, W.L., Grauman, K., Sha, F.: Diverse sequential subset selection for supervised video summarization. In: NIPS. (2014) 3, 6
7. Kulesza, A., Taskar, B.: Determinantal point processes for machine learning. Foundations and Trends in Machine Learning **5**(2–3) (2012) 4, 6
8. Buchbinder, N., Feldman, M., Seffi, J., Schwartz, R.: A tight linear time (1/2)-approximation for unconstrained submodular maximization. SIAM Journal on Computing **44**(5) (2015) 1384–1402 5
9. Gygli, M., Grabner, H., Van Gool, L.: Video summarization by learning submodular mixtures of objectives. In: CVPR. (2015) 5
10. Zhang, K., Chao, W.l., Sha, F., Grauman, K.: Summary transfer: Exemplar-based subset selection for video summarization. In: CVPR. (2016) 5, 6