In Proceedings of the European Conference on Computer Vision (ECCV), 2018

Supplementary Material: Retrospective Encoders for Video Summarization

Ke Zhang[1], Kristen Grauman[2], and Fei Sha[3]

[1] Dept. of Computer Science, U. of Southern California, Los Angeles, CA 90089
[2] Facebook AI Research, 300 W. Sixth St. Austin, TX 78701
[3] Netflix, 5808 Sunset Blvd, Los Angeles, CA 90028
zhang.ke@usc.edu, grauman@fb.com*, fsha@netflix.com**

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

- Section 1: Other implementation details on model training and inference (Section 3.1 and 3.2 in the main text), as well as performance evaluation (Section 4.1 in the main text).
- Section 2: Additional discussions, comparisons and results analysis (Section 4.3 and 4.4 in the main text)
- Section 3: Details of an LSTM cell

1 Implementation Details

1.1 Model Training

Shot boundaries detection As mentioned in section 3 and 4 of the main text, when $\boldsymbol{B}$ is not given (which is the case in the datasets [5, 14, 16] in our experiments), we learn a shot-boundary detection model to infer the boundaries of the shots. In this paper, shot boundaries for each video are predicted by a pre-trained LSTM. By thresholding the predicted confidence for each frame, we get the shot boundaries for each video. The optimal shot boundaries are determined by the summarization performance on the validation set. The average shot durations are 45, 75, and 35 frames for SumMe, TVSum, and VTW respectively. This is also consistent with the prior knowledge that videos of TVSum are much longer than ones of the other two datasets.

Keyshots for training Given the generated shots in section 1.1, our model still needs to know the ground-truth during training, *i.e.* which of them are keyshots, denoted as KEYSHOTS-TRAIN. However, the keyshots annotated by human, denoted as KEYSHOTS-HUMAN, do not necessarily align with the generated shots. For example, one can annotate a keyshot from the 60-th frame to the 80-th frame, but the predicted shots may be $[0, 50], [51, 70]$, and $[71, 110]$, $\cdots$. To construct KEYSHOTS-TRAIN from KEYSHOTS-HUMAN, we first compute the overlapping ratio of each shot w.r.t. KEYSHOTS-HUMAN, which is the overlapping between a

* *On leave from University of Texas at Austin (grauman@cs.utexas.edu)*
** *On leave from U. of Southern California (feisha@usc.edu)*

shot and KEYSHOTS-HUMAN divided by the shot length. We rank all shots in the descending order by their overlapping ratios, and select them in order into KEYSHOTS-TRAIN so that the total duration of KEYSHOTS-TRAIN is below the one of KEYSHOTS-HUMAN. Taking the example above, scores of the first 3 shots are $0, 0.5, 0.25$. So if the total duration of KEYSHOTS-TRAIN is less than the one of KEYSHOTS-HUMAN, we always take $[51, 70]$ prior to $[71, 110]$.

Hyperparameter tuning For different combinations, we tune the hyperparameters by their performances on the validation set, and the optimal hyperparameters are as follows:

- For *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$), $\lambda^* = 0.1$, $\eta^* = 0.15$ on SumMe, $\lambda^* = 0.1$, $\eta^* = 0.1$ on TVSum and $\lambda^* = 0.2$, $\eta^* = 0.2$ on VTW.
- For *re*-SEQ2SEQ ($\lambda = 0, \eta = \eta^*$), $\eta^* = 0.2$ on SumMe, $\eta^* = 0.2$ on TVSum and $\eta^* = 0.25$ on VTW.
- For *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = 0$), $\lambda^* = 0.2$ on SumMe, $\lambda^* = 0.15$ on TVSum and $\lambda^* = 0.3$ on VTW.

Applicability to long videos As in previous works [19, 11], we introduced hierarchical modeling to cope with long videos. The longest videos in our experiments are 10 minutes or more. One level of hierarchy in the encoder can help reduce the amount of inputs in an order of magnitude. For even longer ones, e.g. ego-centric videos [8, 9], we can stack more hierarchies.

1.2 Model Inference

Summary formats Our model generates a sequence of predicted features. Those features do not directly correspond to real video shots. To form a set of the original video keyshots as summary for inference, we find the nearest neighbor shot in the video for each predicted feature $\boldsymbol{y}_l$ on Euclidean distance and add it to the summary. To convert the keyshot to the frame-level importance for evaluation (e.g. required by SumMe [5]), we directly mark frames which are contained in the summary with score 1, and 0 for frames not covered by the summary.

Summary length Note that during inference or testing, the input video is considered as unseen and naturally the model doesn't see the ground-truth summary or know when to stop. In this case, we learn an extra EOS vector standing for the end of summary. We also learn a scalar function END$(\cdot)$ – when END(EOS) is greater than 0.5, we will stop generating the summary (feature vectors). We also learn a begin-of-summary (BOS) vector to initialize the decoder LSTM, with a multilayer perceptron which takes the average of all the feature vectors in the sequence as the inputs.

Computational cost The average inference time for each video in our experiments is $1.2s$ given the frame features. Extracting features would be $2ms$ per frame which can be processed offline. Additionally, the use case can be made while summarized videos are pre-computed and stored.

1.3 Evaluation with User Summaries

Comparison of User Summaries (CUS) [2] is one of the most widely used evaluation metric for video summarization, where the summary of any video in the dataset is annotated by a number of users, i.e. after watching the video, all the users are asked to provide their summaries. Those user summaries are then regarded as the ground-truth and compared with automatically generated summaries by the summarization model(s). Most state-of-the-arts datasets apply CUS and F-score for performance evaluation, including the datasets used in our experiments.

We compute the F-score between the predicted summaries with every human-created summary on both TVSum [14] and SumMe [4, 5]. Then we follow the evaluation protocols proposed in each dataset: average for TVSum [14], and take the maximum for SumMe [4, 5] over the number of human summaries to obtain the F-score for each predicted summary. Note that the authors of SumMe [5] switched from average to maximum in [4]. We hence follow [4] to make a fair comparison with existing works [1, 3, 4, 14, 17–19]. Similar to CUS, Gygli *et al.* [5] propose a leave-one-out strategy to measure the human performance by averaging the F-score of each user's summary to others'. However, this human performance does not necessarily upperbound the performance of a predicted summary, which is typically evaluated by comparing with all users [4, 5, 14].

The major drawback of evaluation with CUS is the time-consuming and labor-intensive annotation procedure, which leads to the limited size of each annotated dataset, mostly less than 100 videos. Supervised methods, in particular deep neural network-based ones, perform well but require a large amount of annotated data [18]. Our model overcomes the challenge by enabling learning from unlabeled videos, thus reducing the need for labeled data.

2 Additional Experiments and Discussions

2.1 Novelty versus Basic Seq2seq Models

Supervised models for summarization explicitly minimize the discrepancy between the prediction and the human summary in terms of *whether a frame/shot is selected or not* [1, 3–5, 14, 17–19]. They do not optimize whether the predicted summary can recover the original video. For unsupervised models [7, 10, 14], while the objective is to reconstruct, this is almost never achievable as the summary is lossy. Our method overcomes this challenge by proposing similarity embedding of the hidden states encoding the original video and the summary respectively, not the raw videos.

Moreover, it is worth noting that the summary and the original video live in two different domains. While conceptually the encoder for the original video can be used to encode the summary, we would need to solve the domain shift/adaptation problem (for video representation) first. Designing another encoder (the retrospective encoder) was perceived as directly solving the problem at hand.

2.2 Semi-supervised Learning with Fewer Annotated Training Videos

We further examine the effectiveness of *re*-SEQ2SEQ on benefiting from unlabeled video data in a more challenging setting: when there are fewer annotated videos available. We pre-train the model with 500 unlabeled videos, and fine-tune the model in the augmented setting. However, we only take a portion of (annotated) training videos for fine-tuning. The rest remains the same, e.g. 10 videos for validation and 10 videos for testing.

The results of using fewer annotated training data are reported in Table 1. In short, consistent with Table 2 in the main text, we see that unlabeled data clearly help improve the summarization performance. In particular, with 10% training videos fewer, the fine-tuned model obtains comparable results to ones in supervised learning, as reported in Table 1 of the main text.

Table 1. F-scores on TVSum [14] in the semi-supervised learning setting. The model is pre-trained on 500 videos and further fine-tuned with *a portion of annotated training videos*.

portion	50%	70%	75%	80%	85%	90%	100%
pre-training	58.4	61.5	62.2	62.8	63.5	64.0	64.4

2.3 Shot Importance Prediction

In practice, most summarization datasets [5, 14] threshold the duration of the summary within 15% of the video length. However, SEQ2SEQ models tend to generate summaries with high precision yet low recall, *i.e.* the total duration of summarized keyshots may be far below the threshold (on average 10.5%), which is also observed in *re*-SEQ2SEQ (on average 12.1%). It is very hard to require the decoder to generate exact length of total keyshots duration, given limited data in various lengths. During inference, we follow the same strategy as in [18] and add in few more keyshots by their importance. The importance of each shot $\boldsymbol{s}_b, b \in \{1, \cdots, \mathsf{B}\}$ is predicted by a function $f_s(\boldsymbol{s}_b)$ as in [19]. In our implementation, $f_s(\cdot)$ is a multilayer perceptron (MLP) with one hidden layer (256-dims) and trained to minimize the difference between the prediction of the input shot $\boldsymbol{s}_b$ and its ground-truth importance, which is 1 for keyshots and 0 otherwise. When the total duration of summarized keyshots by *re*-SEQ2SEQ is below the threshold, we add the remaining shots to the summary in the descending order of their importance by $f_s(\cdot)$ until the threshold is reached, to raise recall.

To specify the contribution of the proposed approach, in Table 2 we compare the performances of using $f_s(\cdot)$ for shot-level importance only (rank shots in the descending order of their importance and select the top ones within the threshold as the summary), denoted as MLP-IMPORTANCE; keyshots only, denoted as *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$) W/O IMPORTANCE and our model, *re*-SEQ2SEQ ($\lambda =$

$\lambda^*, \eta = \eta^*$). In brief, *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$) W/O IMPORTANCE achieves good results on both datasets, but *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$) improves over it with only few shots added to raise recall. Despite that MLP-Importance is used as an baseline in [18] and performs the worst among all three, it still helps to wrap up the summary for a given threshold in *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$).

Table 2. Performances of using importance only (MLP-IMPORTANCE), keyshots only (*re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$) W/O IMPORTANCE), and *re*-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$). See text for more details.

	SumMe	TVSum
MLP-IMPORTANCE	43.0	61.4
re-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$) W/O IMPORTANCE	43.4	62.3
re-SEQ2SEQ ($\lambda = \lambda^*, \eta = \eta^*$)	**44.9**	**63.9**

2.4 Diversity of Generated Summaries

We follow [5] to assess the diversity of generated summaries. The metric is the averaged distance between each shot and its nearest cluster center (keyshot) - the smaller the more diverse. We compare our model to dppLSTM [18] which explicitly promotes diversity and MLP-shot [18] which does not do so explicitly, and the results are shown in Table 3. The marginal difference between ours and dppLSTM is also confirmed by re-training our model with DPP over the summarized shots as in [18] – we obtain a slightly improved F-score of 64.1 versus 63.9 without adding DPP.

Table 3. Diversity assessments of summaries generated by different models (the smaller the more diverse). See text for details.

model	MLP-shot [18]	dppLSTM [18]	our model
diversity assessment	0.49	0.35	0.38

2.5 Comparison with Autoencoders

Besides [10] which we have compared thoroughly in the Table 1 and Table 4 of the main text, Yang *et al.* [15] also apply autoencoders and deep neural networks to their method. However, their work focuses on highlight detection – while it is a form of reducing video data, it is not considered as a summary of the video as the highlight refers to the salient moments while a summary needs to maintain coverage, relevance and diversity. We compared with their model which attains an F-score of 52.3 on TVSum (while ours is 63.9).

2.6 Generic and Query-specific Video Summarization

Sharghi *et al.* [12, 13] study the query-specific video summarization which is a very different task from our focus on generic video summarization. In particular, in their task, queries are required as input to generate the summary, which thus entails different data annotation and evaluation metrics. When our model is compared with theirs on generic video summarization, their model degenerates to [3] (after removing the query-related component as advised by the authors of [3]) and attains an F-score of only 58.0 on TVSum (while ours is 63.9). Furthermore, we consider extending our method to query-based summarization as a future work – note that our datasets do not support that query-specific form of summarization.

3 Long Short-Term Memory (LSTM)

As designed for modeling long-range dependencies, at the core of the LSTMs [6] are memory cells $\boldsymbol{c}$, which at every time step allow the network to learn when to forget previous hidden states and when to update hidden states given new information. The cells are modulated by nonlinear sigmoidal gates, and are applied multiplicatively. The gates in LSTMs also play an important role: the input gate ($\boldsymbol{i}$) controlling to what extent the LSTM considers its current input ($\boldsymbol{x}_t$), the forget gate ($\boldsymbol{f}$) allowing the LSTM to forget its previous memory ($\boldsymbol{c}_t$), and the output gate ($\boldsymbol{o}$) regulating how much of the memory to transfer to the hidden states ($\boldsymbol{h}_t$). To summarize, they jointly collaborate to form the LSTM and in particular make it possible to learn complex long-term dependencies. The recurrences for the LSTM are then defined as:

$$
\begin{aligned}
\boldsymbol{i}_t &= sigmoid(\boldsymbol{W}_i[\boldsymbol{x}_t^{\mathrm{T}}, \boldsymbol{h}_{t-1}^{\mathrm{T}}]^{\mathrm{T}}) \\
\boldsymbol{f}_t &= sigmoid(\boldsymbol{W}_f[\boldsymbol{x}_t^{\mathrm{T}}, \boldsymbol{h}_{t-1}^{\mathrm{T}}]^{\mathrm{T}}) \\
\boldsymbol{o}_t &= sigmoid(\boldsymbol{W}_o[\boldsymbol{x}_t^{\mathrm{T}}, \boldsymbol{h}_{t-1}^{\mathrm{T}}]^{\mathrm{T}}) \\
\boldsymbol{c}_t &= \boldsymbol{i}_t \odot tanh(\boldsymbol{W}_c[\boldsymbol{x}_t^{\mathrm{T}}, \boldsymbol{h}_{t-1}^{\mathrm{T}}]^{\mathrm{T}}) + \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} \\
\boldsymbol{h}_t &= \boldsymbol{o}_t \odot tanh(\boldsymbol{c}_t),
\end{aligned}
\tag{1}
$$

References

1. Chao, W.L., Gong, B., Grauman, K., Sha, F.: Large-margin determinantal point processes. In: UAI (2015)
2. 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), 56–68 (2011)
3. Gong, B., Chao, W.L., Grauman, K., Sha, F.: Diverse sequential subset selection for supervised video summarization. In: NIPS (2014)
4. Gygli, M., Grabner, H., Van Gool, L.: Video summarization by learning submodular mixtures of objectives. In: CVPR (2015)
5. Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: ECCV (2014)
6. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
7. Kim, J., Grauman, K.: Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates. In: CVPR (2009)
8. Lee, Y.J., Ghosh, J., Grauman, K.: Discovering important people and objects for egocentric video summarization. In: CVPR (2012)
9. Lu, Z., Grauman, K.: Story-driven summarization for egocentric video. In: CVPR (2013)
10. Mahasseni, B., Lam, M., Todorovic, S.: Unsupervised video summarization with adversarial lstm networks. In: CVPR (2017)
11. Ng, J.Y.H., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR (2015)
12. Sharghi, A., Gong, B., Shah, M.: Query-focused extractive video summarization. In: ECCV (2016)
13. Sharghi, A., Laurel, J.S., Gong, B.: Query-focused video summarization: Dataset, evaluation, and a memory network based approach. In: CVPR (2017)
14. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: Tvsum: Summarizing web videos using titles. In: CVPR (2015)
15. Yang, H., Wang, B., Lin, S., Wipf, D., Guo, M., Guo, B.: Unsupervised extraction of video highlights via robust recurrent auto-encoders. In: ICCV (2015)
16. Zeng, K.H., Chen, T.H., Niebles, J.C., Sun, M.: Title generation for user generated videos. In: ECCV (2016)
17. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Summary transfer: Exemplar-based subset selection for video summarization. In: CVPR (2016)
18. Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: ECCV (2016)
19. Zhao, B., Li, X., Lu, X.: Hierarchical recurrent neural network for video summarization. In: ACM MM (2017)