# TOWARDS PRACTICAL AND EFFICIENT LONG VIDEO SUMMARY

*Xiaopeng Ke, Boyu Chang, Hao Wu *, Fengyuan Xu, Sheng Zhong*

State Key Laboratory for Novel Software Technology, Nanjing University, China

## ABSTRACT

Recently, video summarization (VS) techniques are widely used to alleviate huge processing pressure brought by numerous long videos. However, it is hard to summarize long videos efficiently since processing hundreds of frames is still time-consuming. In this paper, we find that the Kernel Temporal Segmentation (KTS) method designed for detecting the shot boundaries in SOTA VS methods is time-consuming while handling long videos. To address this issue, we propose the Distribution-based KTS (D-KTS) by fully considering the characteristic of shot length distribution. Furthermore, we propose the Hash-based Adaptive Frame Selection (HAFS) to improve the system performance by fully taking advantage of the temporal locality of long videos. Our experiments present that the proposed D-KTS is 92.70% faster and takes up 90.08% less memory than the baseline KTS method on average.

***Index Terms*—** Long Video Summary, Optimization, KTS

## 1. INTRODUCTION

With the rapid growth of the number of long videos, Video Summary (VS) techniques have got lots of attention [1, 2, 3, 4, 5, 6, 7, 8, 9]. VS methods can output a subset of frames in the input video, and those selected frames can represent the main content of the original video. With the help of the VS methods, we can convert a long video to a summary for efficient storage and better post-processing.

As shown in Fig 1, previous SOTA VS methods [1, 2, 10, 11, 12] always follow the common paradigm with two modules (preprocessing and model inference module). Their preprocessing modules rely on the Kernel Temporal Segmentation [13, 14] (KTS), which is time-consuming under long videos (about 68% of the total processing time). The KTS computes the variances of every interval and applies the dynamic programming technique to get the shot boundaries. To consider every possible interval and obtain precise boundaries, the complexity of KTS is $O(N^2)$, and it becomes the bottleneck for processing long videos. To handle a 10-minute video, it even needs more than 8 hours while using the baseline KTS method.

* Corresponding Author. Email: hao.wu@nju.edu.cn

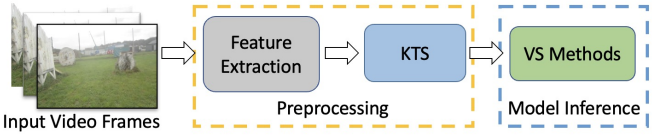


**Fig. 1**: The common paradigm of VS methods. We use DR-DSN [1] on a 476-second video with the KTS [13] and the KTS takes 68% of the total processing time.

In this paper, we design two methods to address the above issue. First, we propose the Distribution-based KTS (D-KTS) method, which exploits that long-tail distribution and selects a proper search range to reduce the complexity. To achieve that optimization, we also present a new dynamic programming algorithm for transferring the baseline KTS to D-KTS. Second, we propose Hash-based Adaptive Frame Selection (HAFS) to pre-process frames and construct an adaptive frame selection mechanism to further improve the pre-processing performance in terms of feature extraction.

In this paper, our contributions are summarized as below:

- We discover the performance bottleneck of SOTA VS methods in the long video scenario, i.e., the baseline KTS makes it impossible to handle long videos practically and efficiently since it considers every possible interval in the video.
- To deal with the issue caused by the baseline KTS, we propose D-KTS after analyzing the shot length distribution. To further reduce the cost of the feature extraction, we propose the Hash-based Adaptive Frame Selection (HAFS) method to skip some frames and maintain the performance.
- We implement our two algorithms and perform comprehensive experiments on two popular VS datasets (TVSum [15] and SumMe [16]) with two representative VS methods (DR-DSN [1] and DSNet [2]). Our results present that our D-KTS outperforms the baseline KTS. Compared to the baseline KTS, our D-KTS is 92.70% faster and saves 90.08% memory.

The rest of this paper is organized as follows. We first describe the overview of the VS pipeline and introduce our D-KTS and HAFS in Section 2. In Section 3, we present some

details of our experiments and discuss the corresponding results. Finally, we make the conclusion at Section 4.

## 2. METHODOLOGY

### 2.1. Overview

As is shown in Figure 2, there are five main steps: (1) Hash-based Adaptive Frame Selection; (2) Feature Extraction; (3) KTS Processing; (4) Getting Frame Scores by VS methods; (5) Summary Generation. At the Feature Extraction step, we apply some popular CNNs (e.g., GoogLeNet [17]) to extract a feature vector for each frame, and we propose HAFS to accelerate that process. Then we utilize the distribution information to reduce the complexity of the KTS and pass those feature vectors and shot boundaries to VS methods. Finally, we obtain the frame score and generate the summary.

### 2.2. Distribution-based KTS

To better describe the algorithm, we first present some notations. Let $V = (x_1, x_2, x_3, ..., x_n)$ represents a $n$-length video where $x_i \in \mathbb{R}^{W \times H \times C}$. $W, H, C$ denote the width, height and channels of a single frame, respectively. $f(\cdot) \in \mathbb{R}^{dim}$ denotes the feature extractor (e.g. The output of the penultimate layer of GoogLeNet) where $dim$ is the dimension of the output feature vector. Then we can obtain the feature vectors of a video as $F(V) = (f(x_1), f(x_2), ..., f(x_n))$. In the baseline KTS method, we have to compute the Gram matrix of feature vectors. We let $K \in \mathbb{R}^{n \times n}$ the Gram matrix and $k_{ij} = g(f(x_i), f(x_j))$ is the element of $K$ where the function $g : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the kernel function.

Our D-KTS method relies on an observation: **The length of shots follows a long tail distribution**. As shown in Figure 3, there is a long tail distribution, and we can model it by using the F-distribution. In the baseline KTS, we need to compute the variance of every possible shot. With the F-distribution, we select an upper bound $u \in \mathbb{R}$ of the shot length to reduce the complexity of the baseline KTS (i.e., We need to compute variances with a sliding window).

In conventional KTS, we need to compute the unnormalized variances like:

$$v_{t,t+d} = \sum_{i=t}^{t+d-1} k_{ii} - \frac{1}{d} \sum_{i,j=t}^{t+d-1} k_{ij} \tag{1}$$

where $t$ is the position of the start frame and $d$ is the duration of the possible segment.

The variance matrix can be easily computed by using the Gram matrix $K$ (e.g., The sum of $k_{ii}$ can be computed by the $\text{diag}(K)$ with the prefix-sum technique.). However, the Gram matrix $K$ becomes sparse if we apply the upper bound, and that matrix still takes a lot of memory. We squeeze that Gram matrix to $T(K) \in \mathbb{R}^{n \times u}$ to reduce the memory. Furthermore, we propose a triangular calculation to compute the second term of the variance (i.e. $\frac{1}{d} \sum_{i,j=t}^{t+d-1} k_{ij}$). $T(K)$ can be written as:

$$T(K) = \begin{pmatrix} k_{11} & k_{12} & k_{13} & ... & k_{1,u} \\ k_{22} & k_{23} & k_{24} & ... & k_{2,u+1} \\ k_{33} & k_{34} & k_{35} & ... & k_{3,u+2} \\ \vdots & \vdots & \vdots & ... & \vdots \end{pmatrix} \tag{2}$$

The first column can be utilized for computing the first term of the variance. To compute the second term, we use the dynamic programming in $T(K)$. For example, if we want to compute the second term of $v_{14}$, $\sum_{i,j=1}^{3} k_{ij} = k_{11} + 2k_{12} + 2k_{13} + k_{22} + 2k_{23} + k_{33}$ can be represented by two triangle $t_1 = (k_{11} + 2k_{12} + k_{22})$ and $t_2 = (k_{22} + 2k_{23} + k_{33})$ with the top term $p_{13} = k_{13}$ and repeat term $k_{22}$. Here, we have $\sum_{i,j=1}^{3} k_{ij} = t_1 + t_2 + 2p_{13} - t_1 \cap t_2$ where the repeat term $t_1 \cap t_2 = k_{22}$ is still a small triangle. In the iterations, we compute the triangle from the smallest thus we reduce some repeat computations. Generally, we let $V(t, t+d-1) = \sum_{i,j=t}^{t+d-1} k_{ij}$ and the formalization can be written as:

$$D_{i,j} = D_{i-1,j} + D_{i-1,j+1} + 2k_{j,i+j-1} - D_{i-2,j+1} \tag{3}$$

where $D_{0,j} = 0, D_{1,j} = k_{jj}, i \geq 2, j \geq 1$. Then we can utilize the equation $V(t, t+d-1) = D_{d,t}$ to compute the second term as $\frac{1}{d} V(t, t+d-1)$.

### 2.3. Hash-based Adaptive Frame Selection

In practice, we also observe that **the visual similarity of two intra-shot frames is higher than that of two inter-shot frames**. To reduce the cost of feature extraction, we propose a hash-based adaptive frame selection method called the HAFS.

For the first frame, we record its hash vector through the d-hash [18]. Then, we compute the hash vector for each frame and make a comparison with the record vector. If the Hamming distance is larger than our threshold, we replace the record with the current hash vector and extract the feature vector through the neural network. Otherwise, we repeat the feature vector of the record frame without making feature extraction on the current frame.

## 3. EXPERIMENTS

### 3.1. Datasets and Implementation Details

**Dataset.** We use SumMe [16] and TVSum [15] datasets for evaluation. SumMe contains 25 videos on different topics and the video lengths vary from 1.5 to 6.5 minutes. Each video has frame-level binary scores annotated by 15 to 18 users. TVSum contains 50 videos selected from 10 categories (i.e., 5 videos per category), and the video lengths vary from 1 to 10 minutes. Each video has shot-level importance scores annotated by 20 users, and each shot is 2 seconds long.

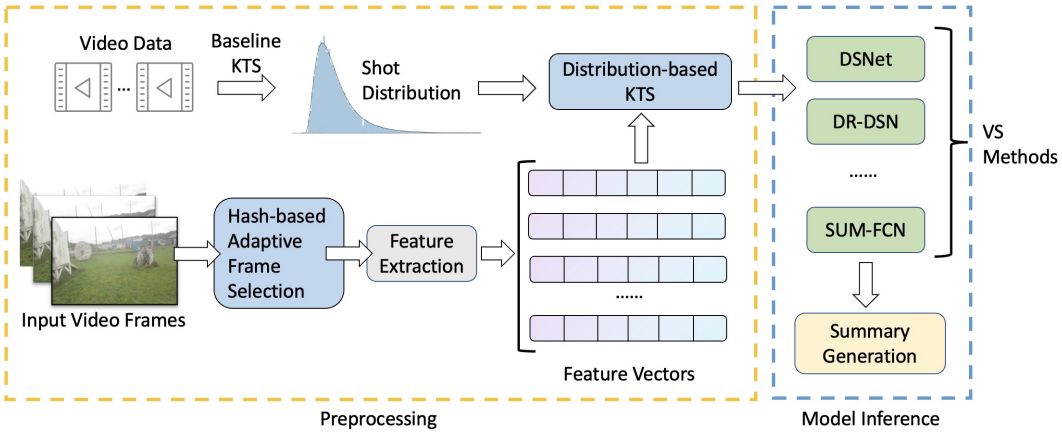


**Fig. 2**: The distribution of segment lengths and the overview of the video summary pipeline with our optimizations.

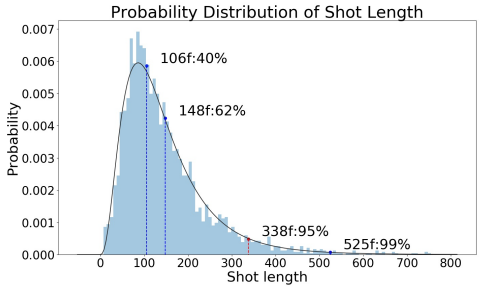


**Fig. 3**: The distribution of the shot length.

**Implementation Details.** The experiments are carried out under python 3.6.13, pytorch 1.4.0, cudatoolkit 10.0.130, opencv 3.4.2, torchvision 0.5.0. The operating system is Ubuntu 16.04. Our device has 4 TITAN Xp 12GB GPUs, Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz, and 126GB memory.

According to our statistics, the shot length distribution of the datasets follows the F distribution, in which 95% of the shot lengths are less than 338 frames as shown in Figure 3. Hence $u$ is assigned to $min(n, 338)$. The maximum change point number $m$ is empirically assigned as $frame\text{-}number/106 * 2$, where 106 is the modal class of the shot length distribution. The hyperparameter $v$ for the penalty term of KTS is assigned to 1 for default. In Hash-based Adaptive Frame Selection, the threshold $t$ is assigned to 4 according to statistics.

For DR-DSN [1], we use the default settings for training. For DSNet [2], we only use the anchor-free version. When the frame rate reaches 15 fps and 30 fps, the model will become too large to train on one GPU, and there is no batch design for the model, so parallel running is infeasible. Hence we use the linear model among the base models provided.

### 3.2. Evaluation Metrics

For evaluation and summary video generation, we follow the scheme in [15, 19]. In order to convert frame-level probability to shot-level machine summary, the 0/1 knapsack algorithm is applied to KTS segments, and the summary video is supposed to be shorter than 15% of the input video. We take the summary of 30 fps as the ground truth. We also use 5-fold cross-validation, and the average F-score is utilized as the metric to compare between the machine summary and the ground truth.

**Table 1**: Time and memory consumption of the baseline KTS and D-KTS

| Video Length (s) | | 60 | 180 | 300 | 420 | 540 |
|---|---|---|---|---|---|---|
| Processing Time (s) | Baseline KTS | 41.24 | 1057.81 | 4800.53 | 13077.74 | 28057.54 |
| | D-KTS | 21.37 | 193.77 | 548.12 | 1052.04 | 1775.32 |
| Memory (MiB) | Baseline KTS | 178.75 | 756.75 | 1998.13 | 3902.92 | 6269.82 |
| | D-KTS | 120.41 | 196.26 | 281.88 | 350.61 | 420.28 |

### 3.3. Performance on Spacetime Consumption and F-score

**Comparison with baseline KTS.** In order to verify the effectiveness of D-KTS, comparison experiments are conducted with the two versions of KTS on randomly generated data of different lengths. As shown in Table 1, as the video length increases, the time and memory consumption of the baseline KTS increases dramatically. On the contrary, the time and memory consumption of the D-KTS has no significant increase. On average, our D-KTS is 92.70% faster than the baseline KTS while saving 90.08% of memory at the same time.

The F-score of the baseline KTS and D-KTS is reported in Figure 4. The F-score of the baseline KTS increases steadily and levels off gradually as the video length increases. While the F-score of the D-KTS fluctuates around 0.92 and 0.96. The average F-score of the D-KTS is 95.03% of the baseline KTS. The optimizations achieve the great improvement of time and memory consumption successfully.

Based on the D-KTS, video summarization systems are implemented, and then experiments are conducted on the systems to explore the influence of different downsampling rates.

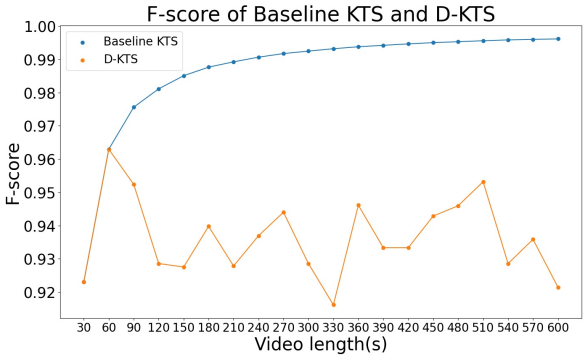


**Fig. 4**: F-score of Baseline KTS and D-KTS

**Influence of the Downsampling Rate.** We implement supervised and unsupervised versions of the video summarization system respectively with DSNet [2] and DR-DSN [1] and analyze the influence of downsampling rates.

Figure 5 shows the F-score of DR-DSN and DSNet on SumMe and TVSum datasets. Basically, frame rate and F-score are positively correlated. Downsampling to 2 fps (used by most related works) has negative impacts on the result. Compared to no downsampling (30 fps), for the TVSum dataset, the F-score decreases 20.92% on average; for the SumMe dataset, the F-score decreases 20.72% on average.

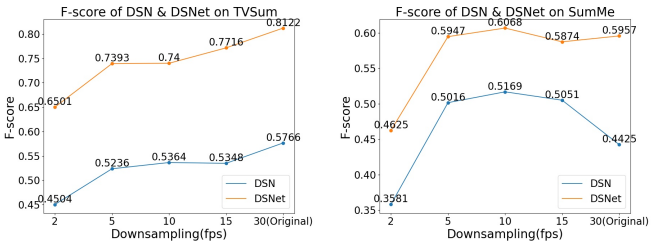


**Fig. 5**: F-score of DR-DSN and DSNet

Figure 6 depicts the proportion of processing time on SumMe and TVSum datasets. The feature is extracted by GoogLeNet-GPU [17]. The time for model inference is very short compared to the others (within 1 second). As the frame rate increases, the time for feature extraction and KTS increases as well, while the time for summary video generation has no obvious change: for SumMe, the time is around 22%; for TVSum, the time is around 5.6%. Videos in the two datasets are within 10 minutes, and our proposed method can process the frame rate up to 15 fps in real-time.

In conclusion, there is a big gap between the F-score of 5 fps and 2 fps, while the processing time of 5 fps is not much more than that of 2 fps. Therefore, taking both F-score and processing time into account, down sampling to 5 fps can achieve a better balance between accuracy and time.

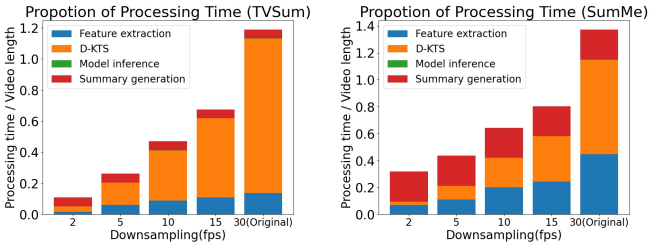


**Fig. 6**: Proportion of processing time on TVSum & SumMe dataset

**Table 2**: Comparison of F-score and average processing time of a single frame for 5 fps

| | TVSum | | SumMe | |
|---|---|---|---|---|
| | **w/o HAFS** | **w/ HAFS** | **w/o HAFS** | **w/ HAFS** |
| **F-score DR-DSN** | 0.5236 | 0.5031 | 0.5016 | 0.4905 |
| **F-score DSNet** | 0.7393 | 0.7197 | 0.5947 | 0.6043 |
| **Feature Extraction (ms)** | 0.95 | 1.46 | 4.15 | 4.24 |
| **KTS (ms)** | 6.14 | 0.16 | 3.62 | 0.16 |
| **KTS + Feature Extraction (ms)** | 7.09 | 1.62 | 7.77 | 4.39 |

**The gain of the Hash-based Adaptive Frame Selection.** As reported in Table 2, after applying the Hash-based Adaptive Frame Selection to feature extraction to D-KTS, the processing time is significantly reduced with a little sacrifice of F-score. For TVSum and SumMe, the time is respectively 79.66% and 44.27% shorter compared to experiments above. And the F-score for DR-DSN and DSNet only decreases 3.06% and 1.04% on average for both datasets.

## 4. CONCLUSION

In this paper, we reveal the performance bottleneck of SOTA VS methods while processing long videos. With the observation of the shot length distribution, we propose D-KTS to reduce the processing time and memory usage. Additionally, we also propose HAFS to accelerate the whole pipeline. To evaluate our optimizations, we conduct comprehensive experiments on two widely-used datasets (TVSum and SumMe) with two popular VS methods (DR-DSN and DSNet). The experiment results show that our D-KTS and HAFS achieve faster processing and less memory usage under long videos with a little sacrifice of accuracy.

## 5. ACKNOWLEDGEMENTS

We sincerely thank the reviewers for their valuable comments helping us to improve this work. This work was supported by National Key Research and Development Program of China 2021YFB3100300, National Natural Science Foundation of China 61872180, Jiangsu “Shuang-Chuang” Program , and Jiangsu “Six-Talent-Peaks” Program.

## 6. REFERENCES


[1] Kaiyang Zhou, Yu Qiao, and Tao Xiang, "Deep reinforcement learning for unsupervised video summarization with diversity-representativeness reward," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018, vol. 32.

[2] Wencheng Zhu, Jiwen Lu, Jiahao Li, and Jie Zhou, "Dsnet: A flexible detect-to-summarize network for video summarization," *IEEE Transactions on Image Processing*, vol. 30, pp. 948–962, 2020.

[3] Mohamed Elfeki and Ali Borji, "Video Summarization via Actionness Ranking," *arXiv:1903.00110 [cs]*, Feb. 2019.

[4] Sijia Cai, Wangmeng Zuo, Larry S. Davis, and Lei Zhang, "Weakly-Supervised Video Summarization Using Variational Encoder-Decoder and Web Prior," in *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., vol. 11218, pp. 193–210. Springer International Publishing, Cham, 2018.

[5] Mingyang Ma, Shaohui Mei, Shuai Wan, Junhui Hou, Zhiyong Wang, and David Dagan Feng, "Video summarization via block sparse dictionary selection," *Neurocomputing*, vol. 378, pp. 197–209, Feb. 2020.

[6] Li Yuan, Francis Eng Hock Tay, Ping Li, and Jiashi Feng, "Unsupervised Video Summarization With Cycle-Consistent Adversarial LSTM Networks," *IEEE Transactions on Multimedia*, vol. 22, no. 10, pp. 2711–2722, Oct. 2020.

[7] Sheng-hua Zhong, Jiaxin Wu, and Jianmin Jiang, "Video summarization via spatio-temporal deep architecture," *Neurocomputing*, vol. 332, pp. 224–235, Mar. 2019.

[8] Mrigank Rochan, Linwei Ye, and Yang Wang, "Video Summarization Using Fully Convolutional Sequence Networks," in *Computer Vision – ECCV 2018*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss, Eds., vol. 11216, pp. 358–374. Springer International Publishing, Cham, 2018.

[9] Yujia Zhang, Xiaodan Liang, Dingwen Zhang, Min Tan, and Eric P. Xing, "Unsupervised Object-Level Video Summarization with Online Motion Auto-Encoder," *arXiv:1801.00543 [cs]*, Aug. 2018.

[10] Yujia Zhang, Michael Kampffmeyer, Xiaoguang Zhao, and Min Tan, "Dtr-gan: Dilated temporal relational adversarial network for video summarization," in *Proceedings of the ACM Turing Celebration Conference-China*, 2019, pp. 1–6.

[11] Zhong Ji, Yuxiao Zhao, Yanwei Pang, Xi Li, and Jungong Han, "Deep attentive video summarization with distribution consistency learning," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 4, pp. 1765–1775, 2020.

[12] Evlampios Apostolidis, Georgios Balaouras, Vasileios Mezaris, and Ioannis Patras, "Combining global and local attention with positional encoding for video summarization," in *2021 IEEE International Symposium on Multimedia (ISM)*, December 2021, pp. 226–234.

[13] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid, "Category-Specific Video Summarization," in *Computer Vision – ECCV 2014*, David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, Eds., vol. 8694, pp. 540–555. Springer International Publishing, Cham, 2014.

[14] Zhuo Lei, Ke Sun, Qian Zhang, and Guoping Qiu, "User video summarization based on joint visual and semantic affinity graph," in *Proceedings of the 2016 ACM Workshop on Vision and Language Integration Meets Multimedia Fusion*, New York, NY, USA, 2016, p. 45–52, Association for Computing Machinery.

[15] Yale Song, Jordi Vallmitjana, Amanda Stent, and Alejandro Jaimes, "Tvsum: Summarizing web videos using titles," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 5179–5187.

[16] Michael Gygli, Helmut Grabner, Hayko Riemenschneider, and Luc Van Gool, "Creating summaries from user videos," in *European conference on computer vision*. Springer, 2014, pp. 505–520.

[17] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[18] Hacker Factor, "The hacker factor blog," January 2013, `http://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html`.

[19] Ke Zhang, Wei-Lun Chao, Fei Sha, and Kristen Grauman, "Video summarization with long short-term memory," in *European conference on computer vision*. Springer, 2016, pp. 766–782.