10-701 Machine Learning Final Project Report: Video Summarization via Deep Convolutional Networks

Chen-Hsuan Lin Wei-Chiu Ma Shih-En Wei

The Robotics Institute
Carnegie Mellon University
{chenhsul,weichium,shihenw}@andrew.cmu.edu

ABSTRACT

As the demand of video summarization techniques increases nowadays, many methods are proposed for how to extract best representating key frames of a video. While most of them rely on hand-crafted image features, we resort to the feature learning power of deep convolutional networks. In this final project, we propose to learn a new image representation such that the similarity of frames are specifically learned for video summarization task, directly supervised by humans key frame selection. To realize this idea, we propose and implement a loss function for deep network in *Caffe*. We also comprehensively studied baseline methods and discuss our qualitative result and properties with them.

1. INTRODUCTION

Nowadays, as the technology in multimedia progresses and the demand for multimedia devices increases, the amount of videos have been increasing rapidly. Due to the exploding amount of data, the need of examining these videos efficiently and effectively becomes more and more urgent. With such demand, video summarization, an efficient management of the raw audio-visual information, has become an evolving research field.

In general, video summarization can be classified into two categories [20]: (i) static storyboard [2, 4] and (ii) dynamic video skimming [13, 19]. Static storyboard can be viewed as key frames selection, while dynamic video skimming is a shorter version of the original video consisting of a series of selected video clips. For both approaches, the appropriate selection of video segments plays a major role in maximizing the diversity and perceptual quality of a video summary. In this work, we address the video summarization problem by focusing on the domain of finding the sequence of frames that *best* represents the content of the original video in a supervised manner. To be more specifc, given all the frames in a video, which of those should be kept automatically to provided concise information about the content, while the essential message of the original video is preserved? Figure 1 gives an example of our goal.

In the past few years, there has been fruitful literature works in computer vision coming up with a variety of methods to solve this problem in an unsupervised manner[9, 12, 15, 16, 17, 25]. They focused on different aspects of videos and proposed a plethora of properties that a good video summary should have for selecting key frames. Gong *et al* [5] summarize those properties and classified them into four categories,

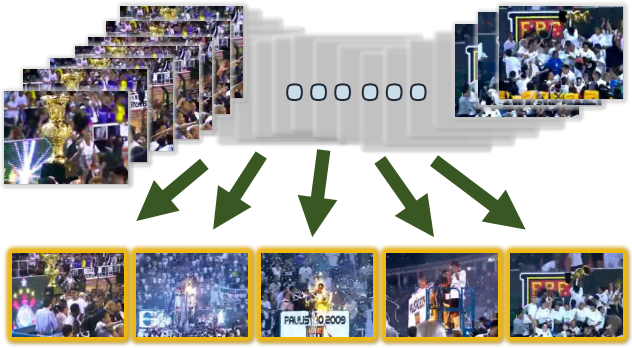

Figure 1: The video summarization task: finding the sequence of frames that best summarizes the original video.

which are *representativeness* (the frames should depict the main contents of the videos) [9, 20, 25], *diversity* (the frames should not be redundant) [15], *interestingness* (the frames should have salient appearance or objects) [17, 20], and *importance* (the frames should contain important objects that drive the visual narrative) [12, 16].

Despite the progress in unsupervised video summarization where the frames are represented by manually crafted features and the criteria for summarization is strictly pre-defined, the video summarization problem is still highly ill-posed. One source of the ill-posedness is the criterion for a good video summarization: the summarization of video is a highly subjective task and it varies a lot based on the favor of different persons. Therefore, it is difficult and inappropriate to manually pre-define any criteria or features of videos to model the summarization decisions of the human. Instead of *designing* the criteria and features, we propose to *learn* the features and criteria used by human for video summarization from labeled videos. In this work, we resort to the power of deep neural networks to learn those features and *imitate* how people make decisions. We expect the deep neural network to operate in the similar way human brains work, making video summarization results match much more closely to the perspectives of humans.

The use of deep neural networks (DNNs) has attracted increasing attention on a variety of applications. With the power of GPU computing, DNNs have outperformed prior

arts in many computer vision tasks such as image classification [11], object detection [22] and structured human pose estimation [23]. However, there hasn't been much reported work focusing on deep learning with videos. Mobahi *et al* [18] demonstrated a deep learning method with sequential data with temporal coherence, particularly video sequences, for object and face recognition tasks. Karpathy *et al* [8] performed large-scale video classification with convolutional neural networks (CNN) and introduced a speed training scheme taking advantage of spatial-temporal information of videos. Despite the appearance of similar recent studies, to the best of our knowledge, there are no technical publications regarding deep learning frameworks tackling tasks related to video summarization. We thus leverage the powerful representation of DNN to learn the the features that human actually rely on to summarize a video.

In our work, we assume that during selecting the key frames, the human uses the *similarity* between pairs of frames rather than the *appearance* of individual frames. Based on this assumption, the *representativeness* and *diversity* of key frames can be estimated using the similarity between pairs of frames and the key frame selection process is recasted as a process where the similarity-based metric is being optmized. Both the features of frames and the key selection procedure are learned from the data rather than pre-defined.

Contributions. The main contribution of our work can be summarized as follows: (i) we introduce the concept of *learning important features* from human-annotated key frames into the field of video summarization, (ii) we implement a novel loss layer using the popular BLVC *Caffe* package [7], (iii) we introduce a novel baseline, which extend the MaxEnt IOC approach [26] from the domain of activity forecasting to video summarization, (iv) we comprehensively survey existing baselines methods and compare our results with them qualitatively.

2. OUR APPROACH

In this section, we first clarify how we transform the video summarization task into a image similarity learning task. Then we state how we learn this similarity by fine-tuning a well-known pre-trained convolutional deep network, with a new loss function and the approximations we applied for differentiability. We also report how we prepare data for training in *Caffe*, and our testing procedure, with some notable detail of the implementation.

2.1 Problem Definition

Given a video consisting of frames $\{I_i | i \in V = \{1, 2, .., N\}\}$, the task of video summarization is to select a set of M key frames $\{I_j | j \in S^\star = \{j_1, j_2, .., j_M\}\}$ to represent this video. Ideally, this selection should capture diverse and representative events in the video with $M \ll N$. In contrast to many existing heuristic ways to select these M frames, which are mostly based on hand-crafted image features, we propose to learn a new image representation that is directly driven by humans key frame selections. To be specific, we aim to learn a image feature space such that the feature points of each video frame are well-clustered in the space, and hence we can easily generate summarization simply by selecting those frames corresponding to the cluster centers and performing clustering algorithm such as k-means.

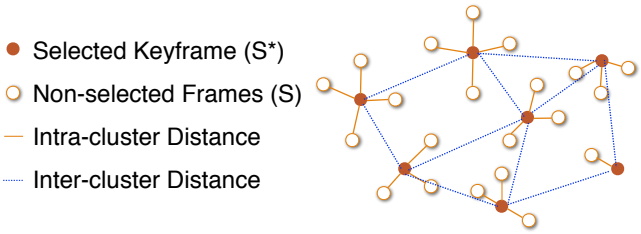

Figure 2: Ideal feature space should minimize intra-cluster distance while keep inter-cluster distance large.

To learn such a feature space, for each training video we want to (i) minimize the distance between every non-selected frames and one of the selected key frames (representativeness), and (ii) keep the distances between selected key frames far apart (diversity). That is, we want to minimize the following loss:

$$L = L_{\text{intra}} + \lambda L_{\text{inter}}, \tag{1}$$

where L_{intra} corresponds to minimizing intra-cluster distances and L_{inter} corresponds to penalizing inter-cluster distances that are too small. Each term will be mathematically defined in the following subsection.

2.2 A Loss Function for Video Summarization

2.2.1 *Minimizing Intra-cluster Distance*

Here we want to minimize the distance between each non-selected frames $S = V - S^\star$ and a key frame in $S^\star$. Note that S and $S^\star$ are disjoint sets. We can thus formulate the loss function as

$$L_{\text{intra}} = \frac{1}{|S|} \sum_{i \in S} \min_{j \in S^\star} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 \tag{2}$$

where $\mathbf{f}_i$ and $\mathbf{f}_j$ is the learned feature of I_i and I_j, and $|S|$ is the number of non-selected frames. Note that we do not make any assumption on how the non-selected frames and the key frames are temporally distributed; we only compare the similarity of two images. Since we are optimizing over the feature representations, the distance between two features can be arbitrarily chosen as long as it is a proper measure of dissimilarity. We chose it to be the squared Euclidean distance (L_2) because the gradients of the loss function can be easily derived and implemented.

Since the minimum function in (2) is not continuously differentiable, we need to find a smooth approximation of the minimum function such that it has a differentiable closed form. Recall that the *softmax function* of a set of real-valued numbers $\{x_1, x_2, \ldots, x_N\}$ maps them to real values in the range of $(0, 1)$, given by

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_{j=1}^{N} \exp(x_j)}$$

This is equivalent to giving the largest x_i a normalized weight much larger than the others and sufficiently close to 1. Inspired by the softmax function, we approximate the mini-

mum function with a "*softmin*" function defined by

$$\text{Softmin}(x_i) = \frac{\exp(\frac{1}{x_i})}{\sum_{j=1}^{N} \exp(\frac{1}{x_j})}$$

which gives the smallest x_i a normalized weight much larger than the others and sufficiently close to 1. The loss function can now be approximated by

$$\begin{aligned} L_{\text{intra}} &= \frac{1}{|S|} \sum_{i \in S} \sum_{j \in S^\star} \|\mathbf{f}_i - \mathbf{f}_j\|_2^2 \frac{\exp(\frac{1}{\|\mathbf{f}_i - \mathbf{f}_j\|_2^2})}{\sum_{k \in S^\star} \exp(\frac{1}{\|\mathbf{f}_i - \mathbf{f}_k\|_2^2})} \\ &= \frac{1}{|S|} \sum_{i \in S} \sum_{j \in S^\star} d_{ij} \frac{s_{ij}}{\sum_{k \in S^\star} s_{ik}} \end{aligned} \tag{3}$$

where the $d_{ij} = \|\mathbf{f}_i - \mathbf{f}_j\|_2^2$ and $s_{ij} = \exp(1/\|\mathbf{f}_i - \mathbf{f}_j\|_2^2)$ are introduced for simplicity. This can be interpretted as follows: the loss of each input frame is its squared Euclidean distance to the closest key frame, which is approximated by the sum of weighted distances, where the weight of closest key frame is the heaviest. We refer to this as the *intra-cluster distance*.

2.2.2 Penalty on Inter-cluster Distance

While minimizing L_{intra}, it is possible that the key frames in $S^\star$ are mapped closer and closer toward each other to make it easier to lower L_{intra}. One extreme case is that all the key frames may be mapped to the same point in the feature space, and obviously all the non-selected frames would be mapped to near the same point. In this case, we cannot separate the distinct frames from the others, and the summarization task would become impossible. Therefore, in additional to minimizing L_{intra} , we also need to add a penalty to avoid key frame features becoming similar to each other.

We formulate the penalty as

$$\begin{aligned} L_{\text{inter}} &= \frac{1}{|S^\star||S^\star - 1|} \sum_{j \in S^\star} \sum_{k \in S^\star \setminus \{j\}} \frac{1}{\|\mathbf{f}_j - \mathbf{f}_k\|_2^2} \\ &= \frac{1}{|S^\star||S^\star - 1|} \sum_{j \in S^\star} \sum_{k \in S^\star \setminus \{j\}} \frac{1}{d_{jk}} \end{aligned}$$

where $|S^\star|$ is the number of key frames. By penalizing the pairwise inverse of distances between key frames, which we refer to as the *inter-cluster distance*, we can ensure the key frames are kept away from each other, since a very small distance between two key frames would result in a huge penalty in that term.

The complete loss function is thus formulated as

$$\begin{aligned} L =& L_{\text{intra}} + \lambda L_{\text{inter}} \\ =& \frac{1}{|S|} \sum_{i \in S} \sum_{j \in S^\star} d_{ij} \frac{s_{ij}}{\sum_{k \in S^\star} s_{ik}} \\ &+ \frac{\lambda}{|S^\star||S^\star - 1|} \sum_{j \in S^\star} \sum_{k \in S^\star \setminus \{j\}} \frac{1}{d_{jk}} \end{aligned} \tag{4}$$

where λ is the penalization constant. This is the target loss function we implement and try to minimize.

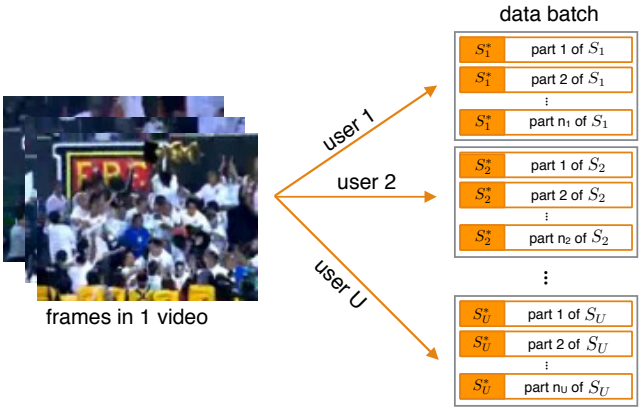

Figure 3: Data preparation for each video. In the training data, each video has multiple (U) users' key frame selections as ground truth.

2.2.3 Gradients for Back-propagation

Given the loss function in Equation (4), we can now derive the gradients of the loss function needed by the back-propagation when training deep networks. We need to consider two cases: (i) how the loss function changes with an non-selected frame feature and (ii) how the loss function changes with a key frame feature.

The gradient of the loss function with respect to an non-selected frame $\mathbf{f}_i$ can be derived by

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{f}_i} =& \frac{2}{|S|} \sum_{j \in S^\star} (\mathbf{f}_i - \mathbf{f}_j) s_{ij} \Big(\sum_{k \in S^\star} s_{ik} \Big)^{-1} \\ & \cdot \left(1 - \frac{1}{d_{ij}} + \Big(\sum_{k \in S^\star} \frac{s_{ik}}{d_{ik}^2} \Big) \Big(\sum_{k \in S^\star} s_{ik} \Big)^{-1} \right) \quad \forall i \in S \end{aligned} \tag{5}$$

Note that the $\partial L_{\text{inter}} / \partial \mathbf{f}_i = 0$, and this gradient depends only on the L_{intra} term, which is its distance from the closest key frame. This is natural since changing $\mathbf{f}_i$ does not affect the inter-cluster distance between key frames.

The gradient of the loss function with respect to a key frame $\mathbf{f}_j$ can be derived by

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{f}_j} =& -\frac{2}{|S^\star|} \sum_{i \in S} (\mathbf{f}_i - \mathbf{f}_j) s_{ij} \Big(\sum_{k \in S^\star} s_{ik} \Big)^{-1} \\ & \cdot \left(1 - \frac{1}{d_{ij}} + \frac{s_{ij}}{d_{ij}} \Big(\sum_{k \in S^\star} s_{ik} \Big)^{-1} \right) \\ & - \frac{2\lambda}{|S^\star||S^\star - 1|} \sum_{k \in S^\star \setminus \{j\}} \frac{\mathbf{f}_j - \mathbf{f}_k}{d_{jk}^2} \quad \forall j \in S^\star \end{aligned} \tag{6}$$

We can see that changing any one of the $\mathbf{f}_j$ affects its distances from all the non-selected frames and all the other key frames. The two gradients fundamentally have different meanings and thus have to be dealt with individually.

2.3 Training in Caffe and Testing Procedure

In this subsection, we introduce how we prepare data for fine-tuning a pre-trained deep convolutional network in *Caffe* as well as our testing procedure.

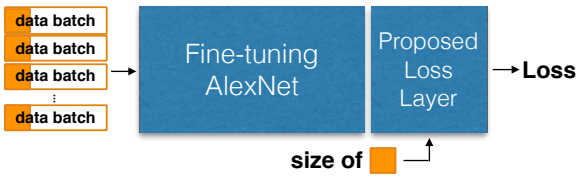

Figure 4: Training architecture

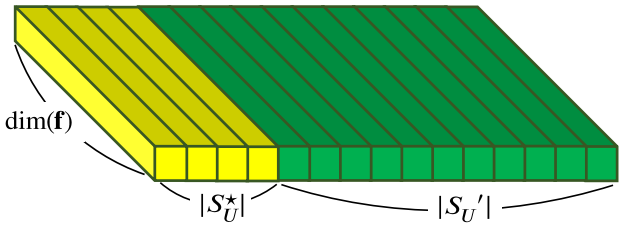

Figure 5: Data structure of an input batch. The features of key frames and non-selected frames are placed together. $|S'_U|$ is the portion size of the non-selected key frames in the batch. In our implementation, $\dim(\mathbf{f}) = 1000$.

Data preparation: Figure 3 shows how we prepare data batches for training. Each video can be associated with multiple key frame selections from different users. Instead of merging them into a single one as oracle summary [5], which loses information, we keep all of these selections as independent ground truth for every video. Our goal is to prepare data batches containing two parts: selected and non-selected, and then the loss and gradient can be calculate within each batch. Note that for a certain user U's selection $S^\star_U$ for a certain video, the number of non-selected frames $|S_U|$ is typically much larger than $|S^\star_U|$. To fit in an acceptable batch size allowed by hardware, we duplicate $S^\star_U$ into multiple data batches, with each followed by only a portion of S_U. We denote this portion a general notation by S'_U, which satisfies $S'_U \subseteq S_U$.

Training by fine-tuning: We cascade our new loss function implemented in *Caffe* to AlexNet [11], and fine-tune it considering the amount of data we have, as shown in Figure 4. Noting that $|S^\star_U|$ can be different for different data batch, instead of providing typical data-label pairs, here the "label" is only used to specify $|S^\star_U|$.

Testing by simply k-means: Given the representativeness and diversity of learned feature from our fine-tuned network, for a test video, we can simply put in all video frames to extract the learned feature of each frame $\mathbf{f}_i$, and easily generate key frames by performing k-means clustering algorithm on them.

2.4 Implementation Details

The structure of the input data batch is organized as shown in Figure 5. As mentioned in the previous subsection, the batch consists of $S^\star_U$, the set of key frames, as well as S'_U, a portion of the set of non-selected frames. The dimension of each data point is denoted by $\dim(\mathbf{f})$, whose value is set to be 1000 in AlexNet.

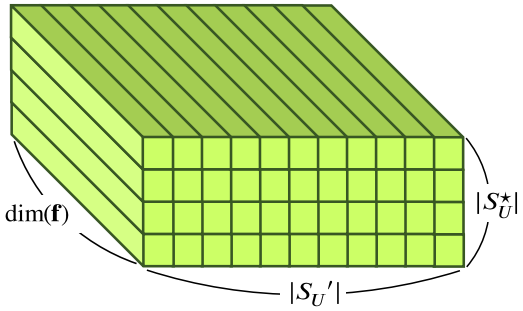

Figure 6: Data structure of a temporary matrix, holding the pairwise feature differences between key frames and non-selected frames.

We can see from (5) and (6) that there are repeating occurrences of $\mathbf{f}_i - \mathbf{f}_j$, d_{ij}, and s_{ij} for the same batch, where $i \in S'_U$ and $j \in S^\star_U$. To avoid recomputing these pairwise information in both forward (loss calculation) and backward (gradient calculation), we store them in temporary matrices of size $|S^\star_U| \times |S'_U|$ after computing in the forward computation. Figure 6 shows how the pairwise $\mathbf{f}_i - \mathbf{f}_j$ are stored.

We would also like to note that using GPUs to compute our loss layer is not as straightforward as other layers like conventional Euclidean distance or convolutional layers in terms of time efficiency, due to the nature of pairwise subtraction. Even we implement it for GPUs, the overhead of data transition makes the parallelization less beneficial. Considering that the loss computation only takes a small portion of overall computation load, we choose to only implement CPU code, and keep other parts of computation parallel on GPU.

We also have to be careful about numerical issues. Recall that our loss function and our gradients all involve computing $s_{ij} = \exp(1/\left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2^2)$. Things could become nasty when $\mathbf{f}_i$ and $\mathbf{f}_j$ are very close to each other, i.e. $\left\|\mathbf{f}_i - \mathbf{f}_j\right\|_2^2$ gets very small. Consequently, s_{ij}, which is the exponential of a large value, can become much larger than the maximum floating number value (in C++ in our implementation). However, we can also observe that every s_{ij} term is coupled with a $\left(\sum_{k\in S^\star} s_{ik}\right)^{-1}$ term, namely the inverse of its summation. When a pair of s_{ij} explodes, it is only natural that $\left(\sum_{k\in S^\star} s_{ik}\right)^{-1}$ also explodes at a same rate, making their product, which is the weight, extremely close to 1. This means that the weights of every other pairs of s_{ik} is extremely close to 0, where $k \in S^\star_U \backslash \{j\}$. Therefore, it is reasonable to just take the hard minimum instead of a weighted sum in this case.

3. EXPERIMENTS

In this section we first report our experiment settings on training deep networks, and introduce others baseline methods for comparison.

3.1 Experiment Settings

We use the VSUMM [3] dataset for both training and testing. From the given 50 videos, we exclude the 10 cartoon videos, and split the dataset into 35 training videos and 5

testing videos. Each video comes with 5 users' selections of key frames. There are a total of around $800K$ videos frames, but we uniformly sampled 1/3 of them for practical considerations, including hard disk space storage and limited training time.

On the settings for training deep network, we train the network on a nVIDIA Titan X GPU. Given its 12GB memory, we set the batch size to 128. On training, we only effectively fine-tune the last 2 fully-connected layers of AlexNet by suppressing the learning rates of previous layers to 1/100 of the current learning rate. On stochastic gradient descent, we set the base learning rate to be 10^{-6} and decrease it to 1/3 of its previous value at the beginning of every epoch, with a total training length of 10 epochs. We set the momentum to 0.9 and weight decay to 0.0005 as typically values. We fix λ in (4) to be 1 throughout the training process.

3.2 Baselines

VSUMM. The first baseline is VSUMM [3], a simple yet effective unsupervised video summarization approach. The whole procedure can be roughly classified into three steps. First, VSUMM calculates the color histogram for each frame in the HSV color space. Second, it performs k-means clustering on those color histograms. At last, VSUMM selects the frames that lie nearest to the centroid of each cluster to be the key frames. Although the idea is straightforward, the performance is surprisingly good, and it has been the state-of-the-art unsupervised approach since 2011[1].

seqDPP. The second baseline is the sequential determinantal point process (seqDPP) [5]. This method not only preserves the characteristic of DPP that the more diverse the frames are, the higher chance they will be selected, but also heeds the inherent sequential structures in video data. We encode each frame with a 8192-dimensional Fisher vector [21] computed from dense SIFT features [14] and then select the frames that maximize the diversity of the selected key frames set. Note that the diversity is calculated by the kernel matrix learned from data. For details of the DPP algorithm, please refer to [5].

MaxEnt IOC. The third baseline is an extension of the maximum entropy inverse optimal control (MaxEnt IOC) framework [26], which have been shown to be very powerful in learning humans' decision policy from observed behavior [10, 6]. Following [10, 6], we use a Markov decision process (MDP) [1] to model humans' strategy in selecting key frames, where the state s_i^t is defined as selecting frame i at time t, and the action $a_{i,j}$ is defined as selecting frame j as key frame right after selecting frame i. As in seqDPP, we encode dense SIFT [14] features into Fisher vectors [21] and use it as a representation for each frame. We learn the transition policy $\pi(a_{i,j}|s_i)$ by maximizing the likelihood of the ground truth key frames annotated by human. For more details, please refer to [26].

3.3 Results

3.3.1 Convergence

We first look into the convergence of training error. Figure 7 shows the objective loss over the number of iterations.

[1]Considering only non task-specific approaches.

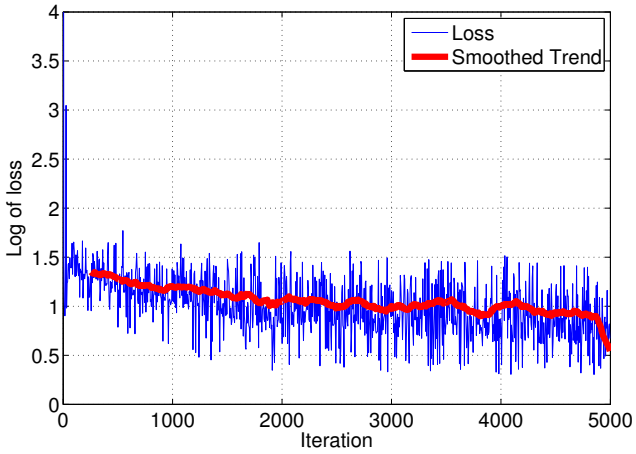

Figure 7: Training error (loss) over time. Blue line: the actual training error. Red line: the decreasing trend of the training error.

We can clearly see that the loss does decrease over time, and it converges around second epoch (3094 iterations per epoch). This implies our setting of 10 epoch is sufficient for the amount of data we use. Plotting Figure 7 not only show how fast it convergences, but it also provides a verification that our implementation on the newly proposed loss function and corresponding gradient is correct.

3.3.2 Visualizing Deep Features

As mentioned in Section 2.2, our deep network learns a feature representation for each frame such that the complete loss function $L = L_{\text{intra}} + \lambda L_{\text{inter}}$ is minimized. As a proof of concept that our feature representation does separate similar frames from dissimilar ones, we use the well-known t-Distributed Stochastic Neighbor Embedding (t-SNE) technique [24] to visualize it in Figure 8. We can see that generally similar frames are clustered together and distinct frames are separated apart. This observation is not only consistent with our expectation, but also demonstrates the reason why we can simply use k-means to classify the frames.

3.3.3 Discussion on the Comparison to Baselines

We compare our summarization results with the user-labeled ground truth and a few other baselines, including VSUMM [3], seqDPP [5], and MaxEnt IOC [26] as mentioned. Figure 9 and 10 shows the selected summary using the above algorithms. We can see that it is arguable that our results capture similar frames as user labelled ones, while other methods are more prone to select replicate scenes (despite they look different). In addition, our method sometimes captures some frames that are not selected by human, but informative in the video.

Besides the quality of summarization, which is from subjective judgement, there are many merits to our proposed framework:

First of all, we argue that our DNN-based feature is more task-specific than hand-crafted features, which tends to lose information. We do not make any assumptions on how the

Figure 9: Summarization result compared to baselines (Video 99 of the VSUMM dataset).

Figure 10: Summarization result compared to MaxEnt IOC [26] (Videos 95, 97, and 98 of the VSUMM dataset).

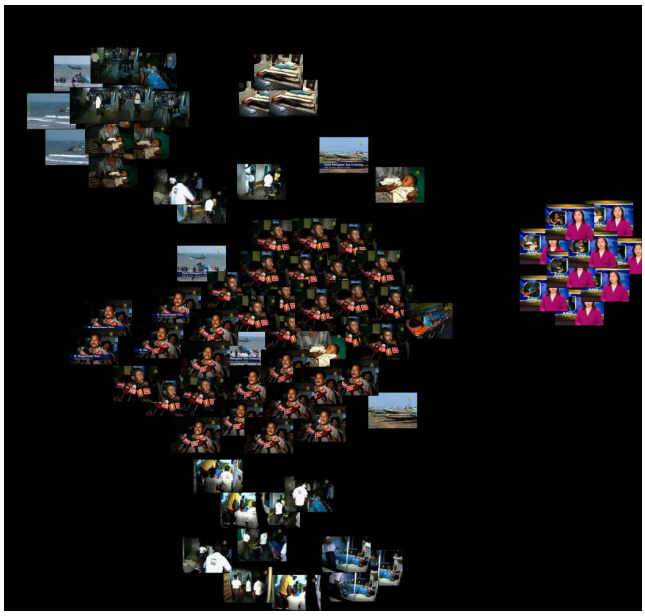

Figure 8: t-SNE visualization of the resulting feature space

feature for video summarization should be designed; the feature is learned solely from data and supported by user studies. There is a probability that the humans' mindset of how a video should be summarized is implicitly taken into account by the supervision from the users' key frame selections. as long as the amount of data is scaled up. As long as the amount of training data scales up, the quality of our result can be improved and become more robust. On the other hand, the use of hand-craft features, such as color histograms in HSV color space adopted in VSUMM [5], is based only on heuristics.

Second, we emphasize the simplicity of our algorithm in the testing phase. Other algorithms, on the contrary, employ complicated methods such as graphical models with Bayesian inferences in seqDPP [5]. This simplicity makes our method scalable to the length of video, whereas the number of probability states in [5] grows quadratically with the length of video.

Finally we would like to address the weakness of our method. The most obviously weakness of our method is that we need large amount of data and long training time, whereas other methods do not have this limitation.

4. CONCLUSION

In this final project we propose a novel idea for video summarization. We learn a new image representation that is good for selecting key frames from videos. Instead of relying on existing or creating heuristic hand-crafted features, we directly learn this image representation from humans' key frame selections through fine-tuning a deep convolutional network. Although we do not have strong evidence that our result is much better than baselines, we discussed trade-offs between our method and baselines. What's more important, we earned valuable experiences in implementing a new layer in the popular deep learning framework *Caffe*, where the experiences make us more capable to attempt on new ideas on deep learning in the future.

5. REFERENCES

[1] R. Bellman. A markovian decision process. Technical report, DTIC Document, 1957.

[2] H. S. Chang, S. Sull, and S. U. Lee. Efficient video indexing scheme for content-based retrieval. *Circuits and Systems for Video Technology, IEEE Transactions on*, 9(8):1269–1279, 1999.

[3] S. E. F. de Avila, A. P. B. Lopes, et al. Vsumm: A mechanism designed to produce static video summaries and a novel evaluation method. *Pattern Recognition Letters*, 32(1):56–68, 2011.

[4] D. DeMenthon, V. Kobla, and D. Doermann. Video summarization by curve simplification. In *Proceedings of the sixth ACM international conference on Multimedia*, pages 211–218. ACM, 1998.

[5] B. Gong, W.-L. Chao, K. Grauman, and F. Sha. Diverse sequential subset selection for supervised video summarization. In *Advances in Neural Information Processing Systems*, pages 2069–2077, 2014.

[6] D.-A. Huang and K. M. Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *Computer Vision–ECCV 2014*, pages 489–504. Springer, 2014.

[7] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[8] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1725–1732, June 2014.

[9] A. Khosla, R. Hamid, C.-J. Lin, and N. Sundaresan. Large-scale video summarization using web-image priors. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2698–2705. IEEE, 2013.

[10] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert. Activity forecasting. In *Computer Vision–ECCV 2012*, pages 201–214. Springer, 2012.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[12] Y. J. Lee, J. Ghosh, and K. Grauman. Discovering important people and objects for egocentric video summarization. In *CVPR*, volume 1, pages 3–2, 2012.

[13] R. W. Lienhart. Dynamic video summarization of home video. In *Electronic Imaging*, pages 378–389. International Society for Optics and Photonics, 1999.

[14] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):978–994, 2011.

[15] T. Liu and J. R. Kender. Optimization algorithms for the selection of key frame sequences of variable length. In *Computer VisionâĂŤECCV 2002*, pages 403–417. Springer, 2002.

[16] Z. Lu and K. Grauman. Story-driven summarization for egocentric video. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2714–2721. IEEE, 2013.

[17] Y.-F. Ma, L. Lu, H.-J. Zhang, and M. Li. A user attention model for video summarization. In *Proceedings of the tenth ACM international conference on Multimedia*, pages 533–542. ACM, 2002.

[18] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 737–744. ACM, 2009.

[19] J. Nam and A. H. Tewfik. Dynamic video summarization and visualization. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 53–56. ACM, 1999.

[20] C.-W. Ngo, Y.-F. Ma, and H. Zhang. Automatic video summarization by graph modeling. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 104–109. IEEE, 2003.

[21] J. Sánchez, F. Perronnin, T. Mensink, and J. Verbeek. Image classification with the fisher vector: Theory and practice. *International journal of computer vision*, 105(3):222–245, 2013.

[22] C. Szegedy, A. Toshev, and D. Erhan. Deep neural networks for object detection. In C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 2553–2561. Curran Associates, Inc., 2013.

[23] J. Tompson, R. Goroshin, A. Jain, Y. LeCun, and C. Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014.

[24] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[25] M. Wang, R. Hong, G. Li, Z.-J. Zha, S. Yan, and T.-S. Chua. Event driven web video summarization by tag localization and key-shot identification. *Multimedia, IEEE Transactions on*, 14(4):975–985, 2012.

[26] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. 2008.