Video Summarization using Neural Networks

DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Technology
in
Computer Science

by

Manish Kumar
[Roll No: CS-1820]

under the guidance of

Nikhil R. Pal
Professor
Electronics and Communication Sciences Unit (ECSU)

Indian Statistical Institute
Kolkata-700108, India

July 2020

To my family and my guide

CERTIFICATE

This is to certify that the dissertation entitled **"Video Summarization using Neural Networks"** submitted by **Manish Kumar** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Nikhil R. Pal
Professor,
Electronics and Communication Sciences Unit (ECSU),
Indian Statistical Institute,
Kolkata-700108, INDIA.

Acknowledgments

I would like to show my highest gratitude to my advisor, *Prof. Nikhil R. Pal*, Electronics and Communication Sciences Unit (ECSU), Indian Statistical Institute, Kolkata, for his guidance and continuous support and encouragement. He has literally motivated me with great insights and innovative ideas.

My deepest thanks to all the teachers of Indian Statistical Institute, who have taught me since the beginning of my academic year here.

Finally, I am very much thankful to my parents and family for their everlasting supports.

Last but not the least, I would like to thank all of my friends for their help and support. I thank all those, whom I have missed out from the above list.

Manish Kumar
Indian Statistical Institute
Kolkata - 700108 , India.

Abstract

With the rapid increase in video data, the need for video summarization has increased. it allows us to search and retrieve video data easily. In this work, we have tried to solve the problem of video summary using unsupervised learning. Our method is based on the extraction of key frames. So we first segmented the shots and extracted key frames from each shot. And these key frames are combined to generate a storyboard. We have experimented with different clustering techniques such as k-mean, k-medoid and self-organizing maps. We evaluated our results using a human-generated summary.

Keywords: *Video summarization, Key frames, Clustering, Self Organizing Maps, K-Mean, K-Medoid.*

Contents

Chapter 1

Introduction

1.1 Introduction

The amount of video content on the Internet has increased exponentially due to the increase in cheap Internet services and video creation tools on smartphones and many more such devices. So the demand for video processing has also increased. And searching over a large number of videos is a challenging task. When a user searches for a video using keywords, the search engine's job is to find meaningful videos so that the retrieved videos match the intention of the user to the extent possible. So here the importance of video summary is very high. Video summarization is a task of making a long video a usable data so that the original meaning of video is not lost and the space occupied to save the video is reduced [18].

Video summary is very important in video content search and for producing meaningful video explanations. Video summarization can be done by humans, but there are millions of videos nowadays, so computer algorithms are the best option to handle the huge amount of videos in video summarization.

There are many techniques to summarize a video. Deterministic algorithms are not a preferred choice for multimedia applications [8]. So machine learning techniques have become popular to process a video [7]. Machine learning techniques are based on the training data set and test data set. In supervised learning we need a labeled training data set to train our model. but in real life, we don't have much labeled video summary dataset. A man-generated video summary is subjective and a cumbersome task. Different people may have different summaries depending on their perspective. Because of this challenge, we preferred the use of unsupervised methods. Here we have used clustering algorithms and self-organizing maps to achieve this goal.

There are many works done on video summarization based on Different techniques. There are supervised methods and also unsupervised methods. In Muhammad et. al [13] authors have used Deep functions for each frame extracted with a CNN along with the memorability score and entropy of the frames to summarize the video. But this takes advantage of an additional pre-trained model that gives a memorability score that adds an extra load to the network and increases processing time. In Otani et. al [14] authors used deep features extracted from CNN for each segment and they used different clustering algorithms. Rochan et. al [16] used a fully conventional sequence network, an extended version of the fully convolutional network used for semantic segmentation. In this they used temporal features of a video to feed the network. Cai et. al [6] used variational autoencoders to encode the semantic space and then used encoder decoder to summarize the video. On the other hand, Zhang et. al [21] used an LSTM to summarize a video that takes care of the interdependence of a frame the next frame. But among these algorithms some are supervised and some are unsupervised methods. In the field of video summary, only a few works on self-organizing maps attempt to cluster the frames in semantic space and attempt to summarize a video. In Rani [15] they used a self-organizing map to summarize a video. But they used a frame's HSV feature in self-organizing maps to cluster video frames.

Our method differs from this method in that we used deep features of a video frame extracted from a CNN to a self-organizing map to cluster the frames.

This report is presented in the following ways. In section 2 we present the theory used behind the work. Section 3 represents the implementation part of the work. In section 4, we have presented the result. Section 5 is about future work and section 6 is conclusions.

Chapter 2

Theory

2.1 Summarisation techniques

There are mainly two kinds of video summarisation techniques [8].

1. Static video summarisation: Static summarisation involves extracting the keyframes from a video. and the summary video is also called storyboard.

2. Dynamic video summarisation: In dynamic video summary we extract a small portion of video which is continuous in time but it shows a small clip of the full video. This is also called video skim.

There are many steps involved in static video summarisation.

- The first step is by subsampling the video frames uniformly or randomly. These frames are later used in advance processes.

- Next step is shot boundary detection and shot segmentation.

- The next step is to extract the key frame of each shot. keyframe extraction is the most important part in every video summarisation technique.

- Next step is to make clusters of different frames. The main idea behind clustering different frames is to make a group of similar frames and extract out a single frame from each group which is a representative of the corresponding group. Clustering can vary according to the features used and the algorithm used. Different features of a frame can be used such as SIFT, colour histogram, motion vectors etc. Or there also can be different clustering algorithms such as hierarchical, k-mean, k-medoid, density based etc.

2.2 Clustering algorithms

Clustering is an unsupervised Machine learning technique which is used when we don't have a labelled data set. There are 2 most famous clustering algorithms

- **K-means algorithm**: It is a partitioning algorithm in which the data set of n objects is partitioned into k different clusters and each cluster is represented by the cluster centre. We initialise k number of cluster centres initially. then we gradually move the cluster centres toward the training data set. when the training is complete then the k cluster centres represent the centres of the training data set. But note that here the final cluster centers may or may not be the training data Points. K-mean clustering tries to minimize the intra-cluster variance which is squared Euclidean distance. Given a set of observations $\{x_1, x_2, ..., x_n\}$ where each observation is m-dimensional. And K-mean clustering trying to cluster into k clusters. S $=\{S_1, \ldots, S_k\}$, be the set of k cluster centers. It aims to minimise within cluster sum of square,

$$\arg\min_S \sum_{i=1}^{k} \sum_{x \in S_i} \|\mathbf{x} - \mu_i\|^2 = \arg\min_S \sum_{i=1}^{k} |S_i| Var(S_i) \tag{2.1}$$

- **K-medoid algorithm** : Both the K-mean and K-medoid algorithms try to partition the data set and minimizes the intra-cluster distance. But the only difference being the final centres are one among the training data points. This is also called PAM(Partitioning Around Medoid) which was proposed in 1987 [19].

2.3 Kohonen's Self Organising Map

The self organising map [11] was developed by Finnish researcher Teovo Kohonen in the 1980s. Self organising map is an artificial neural network which incorporates unsupervised learning. Here it uses the competitive mode of learning instead of error correction learning like in gradient descent based artificial neural networks. It is used to map a high dimensional data point to a low dimensional space, generally 2-D space, so that we can visualise the topology of input data. it has property that it preserves the topological structure of input space to the low dimensional space. hence it is used for dimensionality reduction also. Here the training happens in three stages.

1. **Competition** : In this stage the input vector $\mathbf{x} = [x_1, ..., x_m]^T$ is compared with the weight vector $\mathbf{w}_j = [w_{j1}, ..., w_{jm}]^T, j = 1, 2, ..., l$ And from the best matching criterion, maximizing the inner product $w_j^T x$ is equivalent to minimizing the Euclidean distance between the vectors $\mathbf{x}$ and $\mathbf{w}_j$ as shown by the author in

[12]. But $\mathbf{w}_j$ should be of unit length. Let's i($\mathbf{x}$) denotes the best matching unit of $\mathbf{w}_j$, then its defined as

$$i(\mathbf{x}) = \arg\min_j \|\mathbf{x} - \mathbf{w}_j\|, j \in \mathcal{A} \tag{2.2}$$

where A is the lattice of neurons which we are trying to train using input data. And the best matching unit(BMU) is the weight vector which has the shortest distance from the input vector. This BMU is called a winner neuron.

2. **Adaptation** : The adaptive process for synaptic weight should be self organised. And for it to be self-organized the weight $\mathbf{w}_j$ should change according to the input vector $\mathbf{x}$. For unsupervised learning we use Hebbian postulate but with a little change. We used the forgetting term $g(y_i)\mathbf{w}_j$ to overcome the problem of one directional operation in normal Hebbian rule. Here $\mathbf{w}_j$ is synaptic weight vector of neuron j and $g(y_i)$ is positive scalar function of y. So the change in weight vector $\mathbf{w}$ can be denoted as

$$\delta\mathbf{w}_j = \eta y_i \mathbf{x} - g(y_i)\mathbf{w}_j \tag{2.3}$$

Here η is learning rate parameter for training. We choose $g(y_i) = \eta y_i$ and $y_i = h_{j,i(\mathbf{x})}$. So the above equation becomes

$$\delta\mathbf{w}_j = \eta h_{j,i(\mathbf{x})}(\mathbf{x} - \mathbf{w}_j) \tag{2.4}$$

In discrete-time form we can write this equation as

$$\mathbf{w}_j(n+1) = \mathbf{w}_j(n) + \eta(n) h_{j,i(\mathbf{x})}(n)(\mathbf{x}(n) - \mathbf{w}_j(n)) \tag{2.5}$$

3. **Cooperation** : So here we define the topological neighbourhood of cooperating neurons around the winning neuron. We get biological evidence that the firing neurons excite the neurons which are present in the immediate neighbourhood of them more than Neurons which are further away from it. So we need to define a neighbourhood function such that the nodes near to the winning node are getting more excited than the nodes which are far away from the winning node. Let $h_{j,i}$ denote the neighbourhood function centered around the winning neuron i and passing through all the other neurons, and say neuron j to denote a specific neuron. Lets $d_{j,i}$ denotes the distance between winning neuron i and excited neuron j. Here $h_{j,i}$ is a unimodel function of $d_{j,i}$. It should satisfy following 2 properties.

   - $h_{j,i}$ should be symmetrical around the winner neuron where $d_{j,i} = 0$.
   - $h_{j,i}$ should be monotonically decreasing with increase in $d_{j,i}$ and it should go to zero as distance approaches infinity. This is necessary condition for the network to converge.

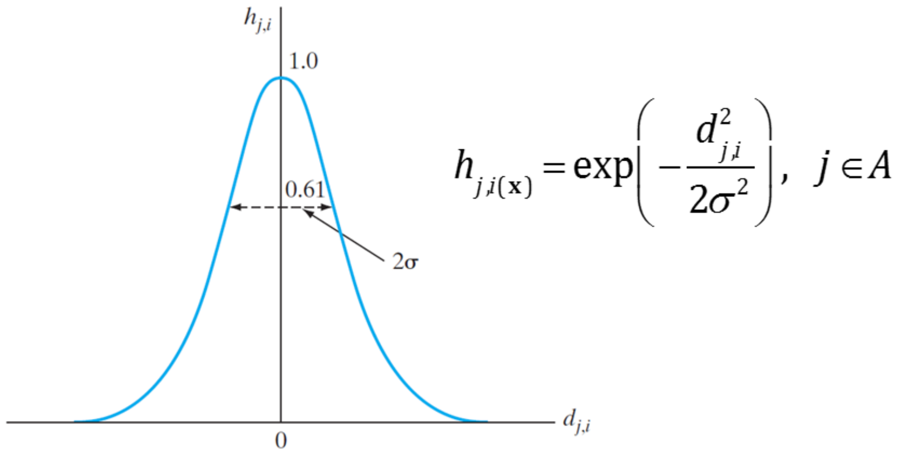


Figure 2.1: Gaussian neighbourhood function, source [12]

The most appropriate function is Gaussian function which is defines as follows

$$h_{j,i(\mathbf{x})} = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2}\right), j \in \mathcal{A} \tag{2.6}$$

From the equation we can see that neighbourhood function $h_{j,i}$ is independent of the choice of winner neuron i. And the σ denotes 'effective width' which enforce the extent to which a neighbour neurons can participate in training process.

In the case of two dimensional lattice, $d_{j,i}$ is defined as

$$d_{j,i}^2 = \|\mathbf{r}_j - \mathbf{r}_i\|^2 \tag{2.7}$$

where $\mathbf{r}_i$ denotes the coordinate of winner neuron and $\mathbf{r}_j$ denotes the coordinate of excited neuron. Another necessary condition for convergence of SOM is size of topological neighbourhood should decrease as the time progresses. This can be achieved by decreasing σ according to the following equation

$$\sigma(n) = \sigma_0 \exp\left(-\frac{n}{\tau_1}\right), n = 0, 1, 2, ... \tag{2.8}$$

This is an exponential decay function. Here σ_0 is the initial width, and τ_1 is the time constant chosen by the user. Hence the final topological neighbourhood function takes the time variant form as shown below

$$h_{j,i(\mathbf{x})}(n) = \exp\left(-\frac{d_{j,i}^2}{2\sigma^2(n)}\right), n = 0, 1, 2, ... \tag{2.9}$$

From the above equation we can see as the n increases, σ decreases, hence topological neighbourhood function decreases in magnitude.

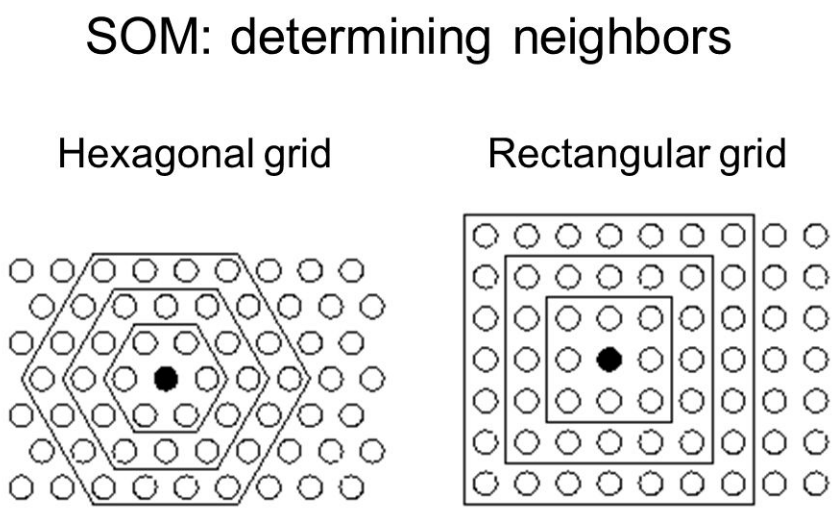


Figure 2.2: Rectangular and hexagonal topological neighbourhood function in discrete setting

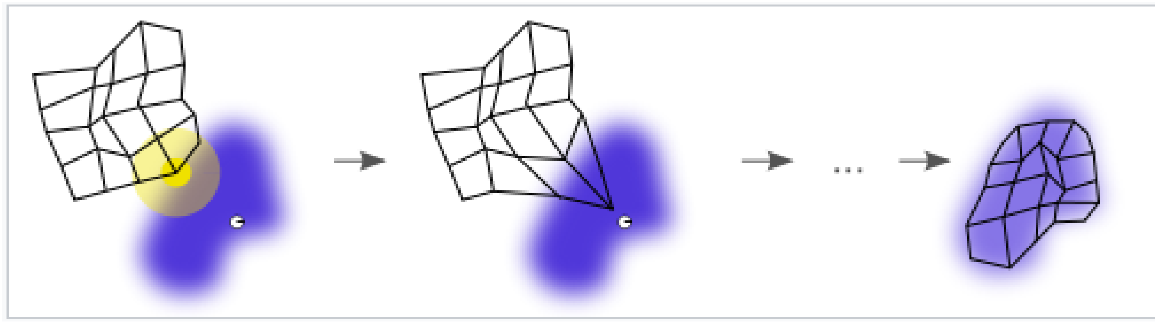


Figure 2.3: In this figure the blue region is the training data set. The grid line is the map which corresponds to the weight vector of the self organising map. in starting the map is distributed randomly in the space. But as the training progresses the map covers the blue origin and approximates the data set in 2-D space. Source Wikipedia [20]

Chapter 3

Implementation

To generate a video summary, we followed many steps to get the best summary of a video. The first step is to sample the video. For sampling, we reduce the number of frames to half or a third according to the length of the video. Then the next step is to pre-process the video. In it, we try to detect and remove the unnecessary parts like the beginning of video and still images. The next step is to detect the scene boundaries. This is done using the Euclidean distance between frames. After detecting the boundary of the scene, we apply clustering algorithms to the remaining frames. We have experimented with different clustering algorithms and with self-organizing maps. Then the resulting frames we got as a summary are presented as storyboards for the user. The full pipeline is shown in the image below.

3.1 Tools and software libraries

The tool we used to complete this project is a combination of several open source software. The software used for data and video conversion preprocessing is FFmpeg.

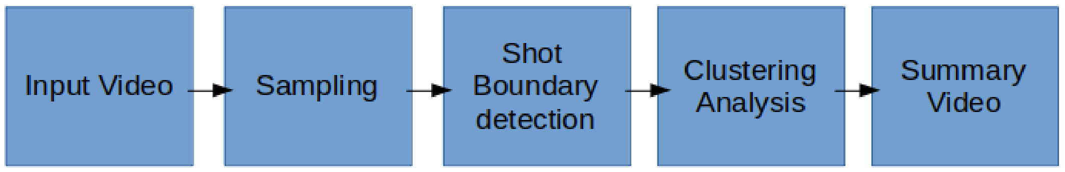


Figure 3.1: Pipeline for proposed video summarization technique

FFmpeg is a multimedia software [1] used for various applications. We used FFmpeg to convert the video format to another and compress the video so that the data we are dealing with is small. We then used several Python libraries to achieve the video summary. The Keras library [3] is used to implement machine learning tools. We used the Keras library to implement a multilayer neural network. We used scikit-learn libraries to implement clustering algorithms. And we used openCV [4] to process images and videos.

3.2 Preprocessing and video formating

The video frame is resized using FFmpeg software. The size of a frame which is acceptable by CNN is 224×224. The preprocessing is done to reduce the size of a video significantly. Because of this we could reduce the processing time of a video significantly.The following command line argument is used to make any video frame size of mp4 format into a video frame size of 224×224.

```
ffmpeg -i input.mp4 -r 30 -s 224x224 output.mp4
```

For compressing any video of any e frame size into the video of single frame size we use the following command line argument.

```
ffmpeg -i input.mp4 -c:v libx264 -crf 30 -c:a copy output.mp4
```

To compress a video of avi video format we use the following command line argument.

```
ffmpeg -i input.avi -vcodec msmpeg4v2 output.avi
```

3.3 Deep features

So what is a deep feature? A deep feature is a representative of video frames which we get when we feed an image to a deep neural network and it passes through all the convolutional layers of the network and it gives an output which is a vector of real numbers. This is a very high dimensional feature vector and this can be projected in a space where similar kinds of images produce feature vectors which are closer to each other in the higher dimensional space. In this way the feature vector contains some semantic meaning of an image.

We have used deep neural networks to extract the feature from each frame of a video. We have used pre-trained googleNet, VGG16, resNet, and mobileNet to extract the

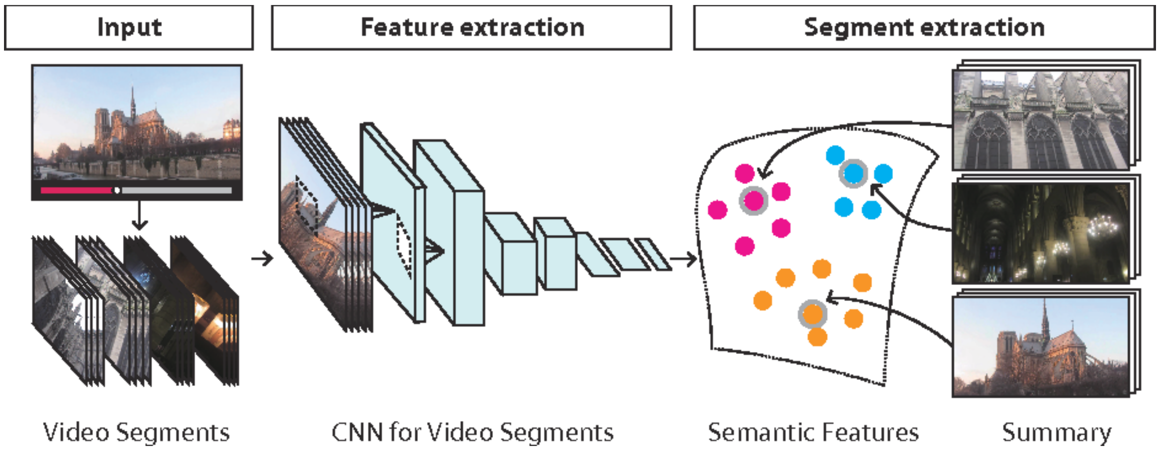


Figure 3.2: Here video is sampled at fixed frame rate and each feature is extracted for each frames using CNN. The frame features are mapped onto semantic space. Cluster centers are the summary frames which are found out after training. Source [14].

features. We experimented with all the neural networks but due to high dimensionality of feature vectors it was taking too much time to process it on clustering algorithms.

VGG16 produces the lowest dimensional feature vector (i.e 25088), which is less than all other nets(googleNet 51200 [17], mobileNet 50176 [10], resnet 100352 [9]), So we have used VGG16 instead of other networks.

3.4 Shot segmentation

Shot segmentation is the next step video summarisation. In a video there can be different scenes . Each scene is different from the other scene in terms of the visual content. These are also called shots. So the shot segmentation is basically to separate different shots so that we can find a keyframe in each shot. And combining different keyframes make a summary of the video. The shot segmentation is performed using the method mentioned in Khan Muhammad et. al [1]. Here the main Idea in Shot segment is to find out unseen frames. So the Euclidean distance between frames has been used.

$$dist = \sqrt{\sum_{i=1}^{n}(\overline{x_i} - x_i)^2, i = 1, 2, \ldots, n} \qquad (3.1)$$

Here n is the length of the feature vector, x_i is the feature vector of the current frame and $\overline{x_i}$ is the feature vector of the previous frame. Then the distance is normalized between 0 and 1. The threshold chosen here is 0.7, which is determined after several number of trials. If the two frames have the distance less than or equal to 0.7, we consider it to be from the same shot. Otherwise, the two frames are considered to be of different shots [13].

3.5 Clustering using K-mean and K-medoid

We have used the sklearn library [5] which is freely available in Python. Here we have to decide how many clusters centers we want to find. Which is equivalent to how many key frames we want to find out for our video summary. So we have chosen a number between 5 to 10. Because most of the user generated summaries in the data set are having video summaries containing frames between 5 and 10.

Two kinds of distance metric is used here In Cartesian coordinate, if $p = (p_1, p_2, ..., p_n)$ and $q = (q_1, q_2, ..., q_n)$ are two points in n-space, then the Euclidean distance $(d(p, q) = \|p - q\|)$ and Manhattan distance $|p_1 - q_1| + \cdots + |p_n - q_n|$ is used as metric in clustering algorithms.

3.5.1 K-mean algorithm

Input: $X = x_1, x_2, x_3, \ldots, x_n$ be the set of n data points, k be the number of clusters

Output: $V = v_1, v_2, \ldots, v_c$ be the set of k centers.

Step1 : Randomly select 'k' distinct cluster centers from the training data.

Step2 : Calculate the distance between each data point and cluster centers.

Step3 : Assign the data point to the cluster center whose distance from the cluster center is the minimum of all the cluster centers.

Step4 : Recalculate the new cluster center using:

$$v_i = \left(\frac{1}{c_i}\right) \sum_{j=1}^{c_i} \mathbf{x}_i \tag{3.2}$$

where, c_i represents the number of data points in i^{th} cluster.

Step5 : Recalculate the distance between each data point and new obtained cluster centers.

Step6 : If no data point was reassigned then stop, otherwise repeat from step 3.

3.5.2 K-medoid algorithm

The most common algorithm of k-medoid clustering is the Partitioning Around Medoids (PAM) algorithm and is as follows:

Step1 : (Initialize) Randomly select k of the n data points as the medoid.

Step2 : (Assignment) Associate each data point to the closest medoid.

Step3 : (Update) For each medoid m and each data point o associated to m swap m and o and compute the total cost of the configuration (that is, the average dissimilarity of o to all the data points associated to m). Select the medoid o with the lowest cost of the configuration.

Step4 : Repeat alternating steps 2 and 3 until there is no change in the assignments.

3.6 Clustering using Self Organizing Maps

We have used a self organising map of Width 6x6. Random initialization was done for the weight vectors . We used a learning rate of 0.01. Initial radius of neighborhood, i.e r_0 was chosen to be $\max(w_{width}, w_{height})/2$. And after each epoch the radius was decreased exponentially by following formula $r = r_0 * \exp\{-epoch/epochs\}$.

3.6.1 Algorithm for SOM

Step 1 : Randomize the node weight vectors in a map

Step 2 : Randomly pick an input vector Dt

Step 3 : Traverse each node in the map

Step 4 : Use the euclidean distance formula to find the similarity between the input vector and the map's node's weight vector

Step 5 : Track the node that produces the smallest distance (this node is the best matching unit, BMU)

Step 6 : Update the weight vectors of the nodes in the neighborhood of the BMU (including the BMU itself) by pulling them closer to the input vector

$$W(t+1) = W(t) + \theta(\mathbf{u}, \mathbf{v}, t).\alpha(t).(D(t) - W(t)) \tag{3.3}$$

Step 7 : Increase t and repeat from step 2 while t $< \lambda$

3.7 Algorithm for video summarization

Input: F = $\{f_1, \ldots, f_n\}$ be the set of n frames of input video.

Output: A subset S $\subset$ F which best represents the summary of the input video

Step1 : Sub sample the video frames uniformly to reduce the number of frames by 1/2, 1/3, or 1/4.

Step2 : Detect shot boundaries and do shot segmentation using Euclidean distance.

step3 : Extract the key frame of each shot using a deep feature obtained from a pretrained VGG16 model.

step4 : Clusters different frames using K-mean, k- medoid and Kohonen's Self organising Maps.

step5 : Do a post processing step to remove redundant and repeating frames.

step6 : Combine all the summary frames in the form of a storyboard.

Chapter 4

Result

4.1 Data

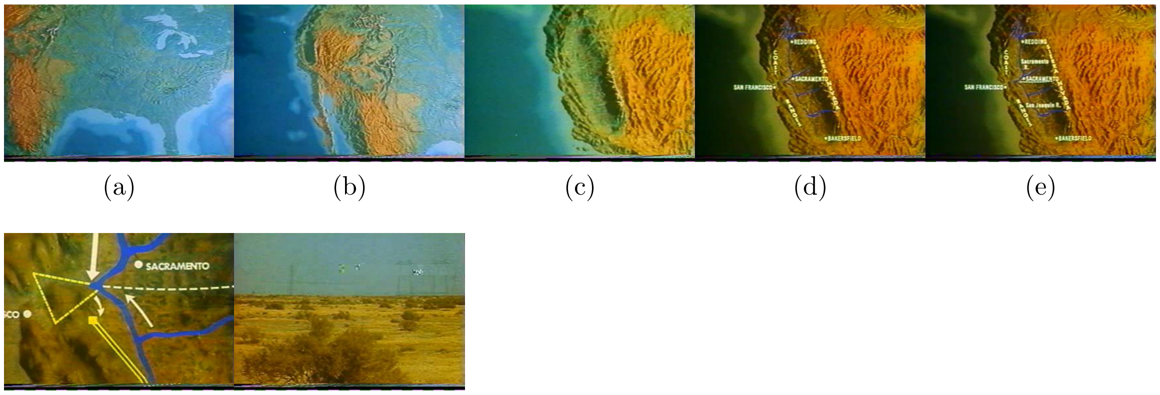
We have used totally two data sets for experimenting with our algorithm.

SumMe : Consisting of 25 videos, each annotated with at least 15 human summaries (390 in total). The data consists of videos, annotations and evaluation code.

VSumm : Title-based Video Summarization (TVSum) data set serves as a benchmark to validate video summarization techniques. It contains 50 videos of various genres (e.g., news, how-to, documentary, vlog, egocentric) and 1,000 annotations of shot-level importance scores obtained via crowd sourcing (20 per video). The video and annotation data permits an automatic evaluation of various video summarization techniques, without having to conduct (expensive) user study.

4.2 Human Annotation

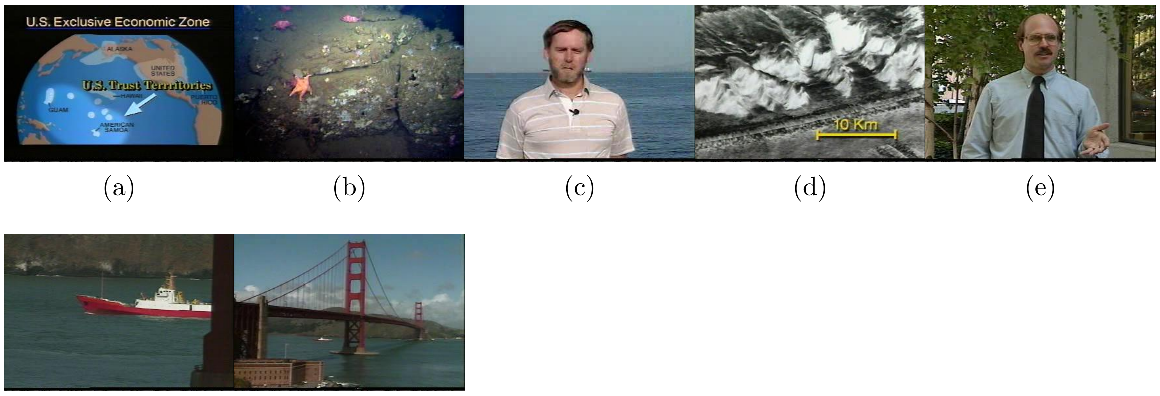
Human annotated video summaries were used to compare the result of our algorithm with the result of human generated summaries. Below are some of the human generated video summaries of VSumm data [2].



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

(f) (g)

Figure 4.1: User summary of video named V-22



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

(f) (g)

Figure 4.2: User summary of video named V-50

(a)

(b)

(c)

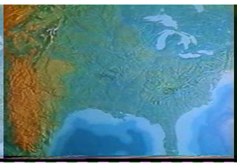
(d)

(e)

(f)

(g)

Figure 4.3: Summary of V-22 using K-mean

(a)

(b)

(c)

(d)

(e)

(f)

(g)

Figure 4.4: Summary of V-22 using K-medoid

4.3 Result and Analysis

As we can see, K-mean and K-medoid are always giving us similar results. But SOM is giving slightly better results than both the other clustering methods.

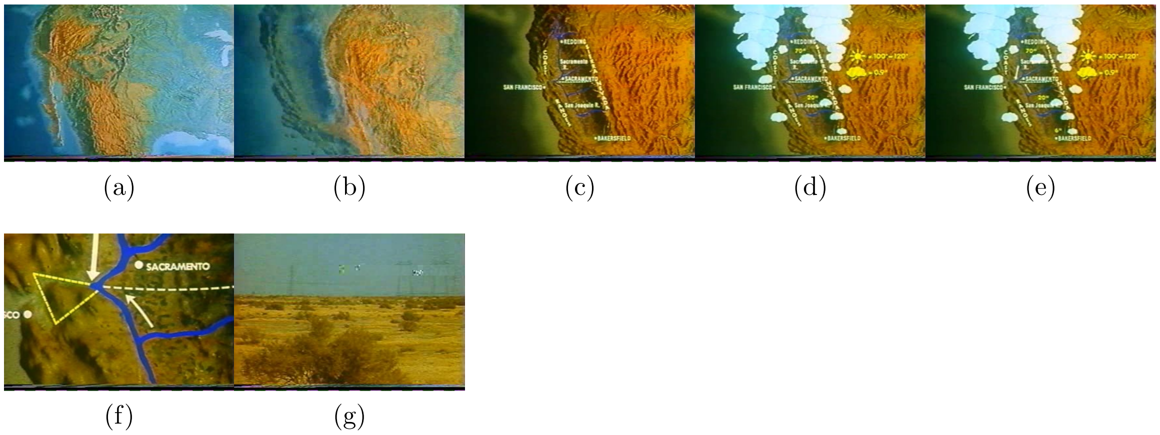


Figure 4.5: Summary of V-22 using SOM

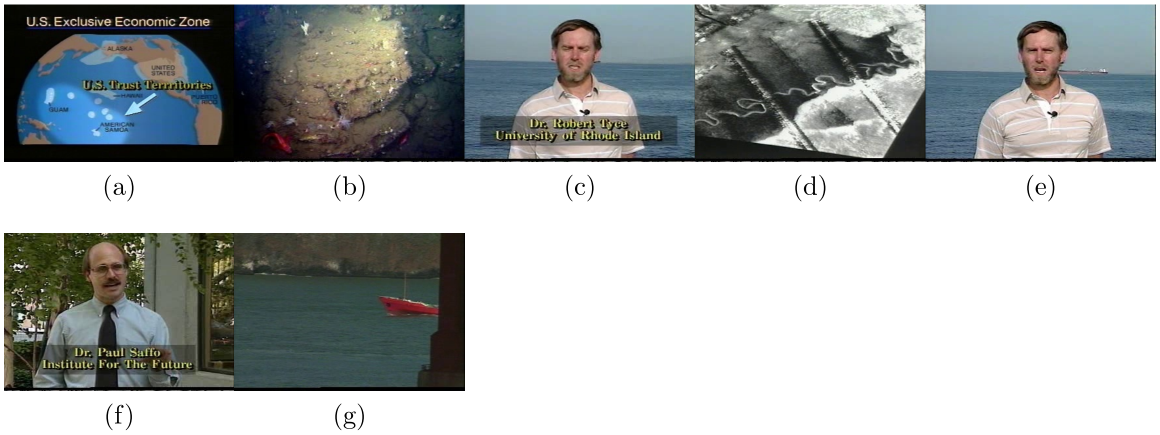


Figure 4.6: Summary of V-50 using K-mean

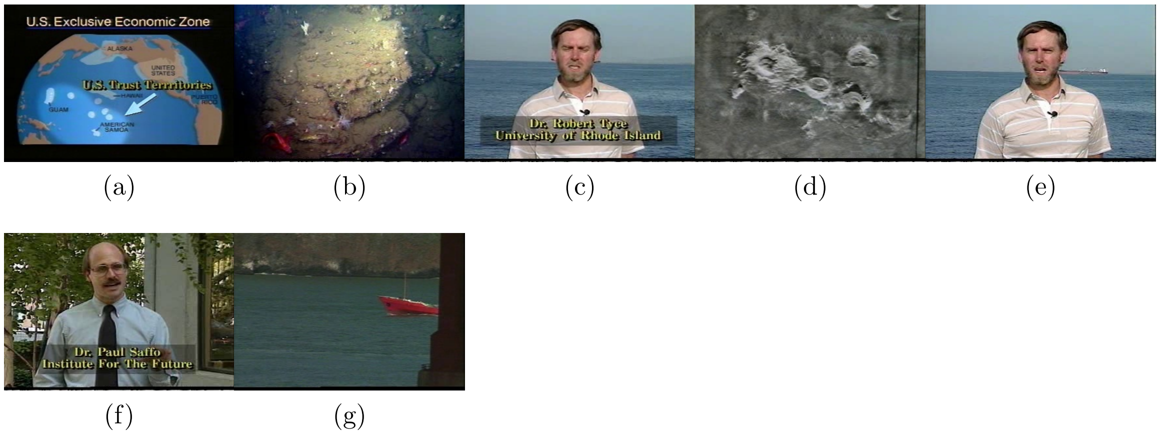
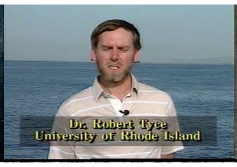


Figure 4.7: Summary of V-50 using K-medoid

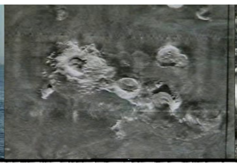
U.S. Exclusive Economic Zone
ALASKA
UNITED STATES
U.S. Trust Territories
HAWAII
GUAM
AMERICAN SAMOA

(a)

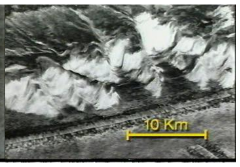
(b)



(c)



(d)



(e)

(f)

(g)

Figure 4.8: Summary of V-50 using SOM

Chapter 5

Future Work

The test data used contains videos of length ranging from 2 minute to 6 minutes. This allows the data containing only less variations in the seen change But in actual practical videos we have video of length more than 6 minutes which contains more variation so our training result is based on only small videos. So it is not generalized for all kinds of video.

Parameters of all the neural networks can be fine tuned to make the video summarization more accurate. We can improve the graphical user interface so that we can visualise the cluster formation of the video frames in the space along with the frame content.

Chapter 6

Conclusion

Video summarization was done on two publicly available datasets that is VSumm and SumMe. We used different clustering algorithms like K-Mean, K-Medoid and Kohonen's Self Organising Maps. We found out that SOM was performing better than the other two clustering algorithms. SOM was able to form a good summary comparable to human annotated summaries.

Although our method was not able to perform at the level of state-of-the-art methods, it was performing quite good with such simple algorithms. With little more effort we can increase the quality of video summarization. This showed we don't need sophisticated techniques but a simple and effective algorithm.

Bibliography

[1] Ffmpeg, https://ffmpeg.org/

[2] https://www.sites.google.com/site/vsummsite/download

[3] Keras library, https://keras.io/

[4] opencv, https://opencv.org/

[5] sk-learn, https://scikit-learn.org/stable/

[6] Cai, S., Zuo, W., Davis, L.S., Zhang, L.: Weakly-supervised video summarization using variational encoder-decoder and web prior. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 184–200 (2018)

[7] Camastra, F., Vinciarelli, A.: Machine learning for audio, image and video analysis: theory and applications. Springer (2015)

[8] Elkhattabi, Z., Tabii, Y., Benkaddour, A.: Video summarization: techniques and applications. International Journal of Computer and Information Engineering 9(4), 928–933 (2015)

[9] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[10] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[11] Kohonen, T.: Self-organized formation of topologically correct feature maps. Biological cybernetics 43(1), 59–69 (1982)

[12] Kubat, M.: Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. The Knowledge Engineering Review 13(4), 409–412 (1999)

[13] Muhammad, K., Hussain, T., Baik, S.W.: Efficient cnn based summarization of surveillance videos for resource-constrained devices. Pattern Recognition Letters 130, 370–375 (2020)

[14] Otani, M., Nakashima, Y., Rahtu, E., Heikkilä, J., Yokoya, N.: Video summarization using deep semantic features. In: Asian Conference on Computer Vision. pp. 361–377. Springer (2016)

[15] Rani, S., Kumar, M.: Social media video summarization using multi-visual features and kohnen's self organizing map. Information Processing & Management 57, 102190 (2020)

[16] Rochan, M., Ye, L., Wang, Y.: Video summarization using fully convolutional sequence networks. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 347–363 (2018)

[17] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1–9 (2015)

[18] Wikipedia contributors: Automatic summarization — Wikipedia, the free encyclopedia (2020), https://en.wikipedia.org/w/index.php?title=Automatic_summarization&oldid=963937331, [Online; accessed 10-July-2020]

[19] Wikipedia contributors: K-medoids — Wikipedia, the free encyclopedia (2020), https://en.wikipedia.org/w/index.php?title=K-medoids&oldid=962166404, [Online; accessed 10-July-2020]

[20] Wikipedia contributors: Self-organizing map — Wikipedia, the free encyclopedia (2020), https://en.wikipedia.org/w/index.php?title=Self-organizing_map&oldid=965486879, [Online; accessed 10-July-2020]

[21] Zhang, K., Chao, W.L., Sha, F., Grauman, K.: Video summarization with long short-term memory. In: European conference on computer vision. pp. 766–782. Springer (2016)