Video Summarization

M.Tech Project Report

Submitted in partial fulfillment of requirements for the degree of

Master of Technology

by

Kurian Jacob
Roll No : 163050039

under the guidance of

Prof. Suyash Awate

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay

June 25, 2018

Dissertation Approval

This dissertation entitled **"Video Summarization"**, submitted by **Kurian Jacob (Roll No: 163050039)** is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology Bombay**.

Prof. Suyash Awate
Dept. of CSE, IIT Bombay
Supervisor

Prof. Ganesh Ramakrishnan
Dept. of CSE, IIT Bombay
Internal Examiner

Prof. Preethi Jyothi
Dept. of CSE, IIT Bombay
Internal Examiner

Declaration

I declare that this written submission represents my ideas in my own words and where others' ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Date: 28th June, 2018
Place: IIT Bombay

Kurian Jacob
Roll No: 163050039

Acknowledgments

I would like to thank my supervisor Prof. Suyash Awate, Prof. Ganesh Ramakrishnan and Prof. Preethi Jyothi for their consistent guidance and support throughout the project. Their suggestions to the issues that we were facing not only helped us to resolve them but also improved our understanding. I would also like to thank my team member Harsh Rathee, insightful discussions with whom helped me better understand the pitfalls of my designs and how to rectify them.

Abstract

Due to the popularity of cameras and video capturing devices and video sharing websites such as YouTube, the amount of video generated per day is huge. The problem is to organize these videos and find out relevant content with minimum effort. Currently, finding the relevant content is difficult because one has to skim through the whole video to capture anything interesting. We seek to find out efficient methods that do this task automatically, that is to reject non-informative content and produce a short logical skim of small duration that captures the gist of story the video wants to convey. In this report, we investigate some of the popular methods in literature for video summarisation, and to improve such methods.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Report Organization . . . 1

2 Background **2**
- 2.1 Feature descriptors . . . 2
- 2.2 Kernel Temporal Segmentation . . . 2
- 2.3 VideoSummy . . . 3
- 2.4 Dictionary learning technique . . . 4
- 2.5 Video segment classification . . . 5
  - 2.5.1 Neural Network as features . . . 5
  - 2.5.2 YOLO v2 . . . 5
- 2.6 Auto encoder . . . 6
- 2.7 Shot boundary detection . . . 6
- 2.8 A naive way to summarise videos . . . 7
- 2.9 LSTM . . . 7

3 Datasets **8**
- 3.1 Event Dataset . . . 8
- 3.2 Toy dataset based on MNIST . . . 8
- 3.3 Football dataset . . . 8

4 Results **10**
- 4.1 Effectiveness of CENTRIST feature descriptor . . . 10
  - 4.1.1 Reconstructing the original image from the CENTRIST descriptor . . . 10
  - 4.1.2 Using the CENTRIST feature descriptor in classification . . . 10
  - 4.1.3 Using the CENTRIST feature descriptor to measure similarity between frames . . . 10
- 4.2 Effectiveness of the Auto-encoder features . . . 10
- 4.3 Kernel Temporal Segmentation . . . 12
- 4.4 Shot Boundary Detection . . . 12
  - 4.4.1 Mahanalobis analysis of the automatically prepared dataset . . . 12
  - 4.4.2 CENTRIST . . . 12
  - 4.4.3 VGGnet feature descriptor . . . 13
  - 4.4.4 Autoencoder features . . . 13
  - 4.4.5 Using the frames themselves as the features . . . 13

4.5 Dictionary based method 16
4.6 Videosummy 16
4.7 Neural Network as Features 17
4.8 Using the naive summarization method 17

5 Future Work **21**

6 Conclusion **22**

List of Figures

2.1 The scheme for the Census Transform. Source: `http://www.mdpi.com/computers/computers-04-00265/article_deploy/html/images/computers-04-00265-g005.png` 2
2.2 The YOLO network can not only predict object in the scene, but it can also given the bounding box location of the object. In our setup , YOLO could process each frame in about 0.7s on a NVIDIA Quadro K20 GPU. 5

3.1 The frames of video created from MNIST database 8

4.1 The similarity across frames for the MNIST video. The (i, j) pixel denote the similarity of the frame i and the frame j. Lighter colors indicate that the frames are closer. 11
4.2 The top row shows the original images, while the second row shows the pictures reconstructed by the fully connected network. 11
4.3 The top row shows the original images, while the second row shows the pictures reconstructed by the convolutional network. 11
4.4 The autoencoder in action. 12
4.5 The cumulative histogram. If we put a threshold to separate datapoints, the separation could be set to a Mahanalobis distance of 4-5. This indicates that a simple classifier such as SVM should work properly for this task. 13
4.6 The gram matrix for the CENTRIST feature descriptor. The first quadrant is the correlation between positives and negatives, second quadrant shows positives and positives, third shows negatives and positives, the fourth negatives and negatives. The negative examples are clustered. 14
4.7 The gram matrix for the Autoencoder feature descriptor. The first quadrant is the correlation between positives and negatives, second quadrant shows positives and positives, third shows negatives and positives, the fourth negatives and negatives. The classes has no recognizable correlation. 15
4.8 Shot detection in action. The timelines are shown in the bottom of the video. The top row shows the predicted shot boundaries, while the bottom row indicates the ground truth. The bright slider indicates the current frame. 16
4.9 The results for validation after bootstrap sampling. There are 20 curves corresponding to leave one out validation of 20 bootstrap samples. The median is shown in the red color. 17
4.10 A boxplot of accuracies vs K-values for leave-one-out validation for different sample sizes. The accuracies range from 50-60%. 18

4.11 A boxplot for the accuracies produced by the poly SVM of order 3. The gamma parameter is $\frac{1}{n}$ where n is the dataset size. The X-axis has results for different sample sizes, 50,100, 150, 200 and 240. The accuracy is found to increase with the increase in training data set. 18

List of Tables

4.1 Results for CENTRIST for shot boundary detection 13
4.2 Results for VGGnet for shot boundary detection 13
4.3 Results for using frames themselves as the features for shot boundary detection . 14
4.4 Results for Summarisation algorithm 20

Chapter 1

Introduction

1.1 Motivation

With the advent of cheap cameras and video capture devices, the amount of digital videos generated each day has grown to enormous proportions. Video sharing websites such as Youtube have added to this growth by making sharing of these videos an extremely simple task. But, it is often a pain to find the right content from within these videos. So, what we seek is to find a method to extract maximum information from these videos with minimum effort. These can be achieved by automatically removing the shots that are relatively unimportant in the video. This is often a difficult task. Most of the time this requires knowledge of the type of video that we are trying to summarise. For example, in a birthday video, the candle lighting and cake cutting can be considered as important. The algorithm is supposed to extract such frames from the video. For some other cases, say a CCTV footage, the important shots might be suspicious activities like people doing abnormal things or deserted baggages. We must note the contrast between these two. One seeks to maximize amount of normal events, while the other tries to minimise it. So, in most of the cases, the task of video summarisation is ambiguous. A good video summary conveys the story of the entire video in the smallest possible time. In this report, we will try to examine the most common methods to summarise videos.

1.2 Report Organization

We begin the report with preliminary ideas about neural networks and segmentation in chapter 2. In addition to techniques for video summarization, we require good annotated data for training, which are introduced in chapter 3. We present the Football match dataset that we made from different video sharing websites to be used for performing general experiments.
In the same chapter 2, we dive into different models currently being used for video summarization. We first give basic principles of and then present different existing models that we surveyed. In chapter 4, we took up the task of implementation and explain various experiments conducted by us and the results obtained. In chapter 5, we present the flaws in the current systems and propose solutions that will help mitigate these.

Chapter 2

Background

2.1 Feature descriptors

We need to define some feature descriptors that could aid us in video summarization subproblems such as scene classification. Here we investigate about the *census transform histogram* short for $CENTRIST$[6], a feature descriptor for scene categorization. Let $I(x, y)$ denote the input image. Then the census transform $C(x, y)$ is computed by looking at the 8-neighbors of (x, y). The $C(x, y)$ value lies between 0 and 255, which has 8 bits in its binary representation. Each of the bit corresponds to one of the 8-neighbors of the pixel under consideration. If the neighboring pixel is lower than the value of the pixel, that corresponding bit is set. If it is not, then the bit is unset.

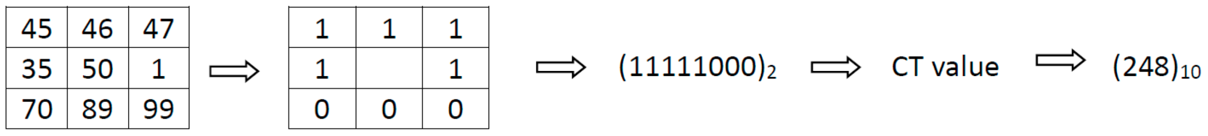

Figure 2.1: The scheme for the Census Transform. Source: `http://www.mdpi.com/computers/computers-04-00265/article_deploy/html/images/computers-04-00265-g005.png`

Now, the $CENTRIST$ is the histogram of this resulting greyscale image.
The $CENTRIST$ feature descriptor has the remarkable property that often the original patch that generated the histogram can be reconstructed by guiding the objective function with the simulated annealing algorithm. This could mean that CENTRIST while taking the descriptor does not destroy the entire information.

2.2 Kernel Temporal Segmentation

[4] Segmentation is important for algorithms that output a skim of the given video. The summarization problem can be reduced to the selection of subsets of the video segments that carries the most amount of information.
This is done through *change point* detection. Change point detection is different from boundary detection of shots. Change point detection attempts find abrupt changes in a 1-D signal. The given signal may be corrupted with noise. If the video is given by a sequence of frame descriptors

$x \in X$, we define a kernel function which captures the similarity of each frame in another feature space. Our aim is to partition the video into m segments such that the in-between variances between frames in any given shot is minimised. This automatically implies that as the number of segments go up, the objective function decreases. To prevent this tendency, we introduce a penalty $g(m,n)$, where m is the number of change points and n is the total number of frames in the video.
The objective function to be minimised is given by

$$J(m,n) = L(m,n) + C \cdot g(m,n)$$

Algorithm 1. Kernel temporal segmentation

Input: temporal sequence of descriptors $\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}$	Cost
1. Compute the Gram matrix A : $a_{i,j} = K(\mathbf{x}_i, \mathbf{x}_j)$	$dn^2/2$
2. Compute cumulative sums of A	n^2
3. Compute unnormalized variances $v_{t,t+d} = \sum_{i=t}^{t+d-1} a_{i,i} - \frac{1}{d} \sum_{i,j=t}^{t+d-1} a_{i,j}$ $t = 0, \ldots, n-1, \quad d = 1, \ldots, n-t$	$2n^2$
4. Do the forward pass of the dynamic programming algorithm $L_{i,j} = \min_{t=i,\ldots,j-1} \left(L_{i-1,t} + v_{t,j} \right), \quad L_{0,j} = v_{0,j}$ $i = 1, \ldots, m_{\max}, \quad j = 1, \ldots, n$	$2m_{\max}n^2$
5. Select the optimal number of change points $m^\star = \arg\min_{m=0,\ldots,m_{\max}} L_{m,n} + Cg(m,n)$	$2m_{\max}$
6. Find change-point positions by backtracking $t_{m^\star} = n, \quad t_{i-1} = \arg\min_t \left(L_{i-1,t} + v_{t,t_i} \right)$ $i = m^\star, \ldots, 1$	$2m^\star$
Output: Change-point positions $t_0, \ldots, t_{m^\star-1}$	

In this equation, $J(m,n)$ denotes the objective function to be minimised, $L(m,n)$ is the sum of all within-shot variances of the frames.
Computation is done with a dynamic programming paradigm. We begin with computing the kernel for each pair of frames. For all segments that begin at a given timeframe t, for a possible duration d, the segment variances can be computed, by looking up the kernel value table. Now, using the dynamic programming approach, $J(i,j)$ can be calculated for all feasible i and j.

2.3 VideoSummy

This is an already existing video summarization software. This program works by dividing the input video into segments of fixed size. Then the software transforms each of these segments into a set of colour histograms. Thereafter, the importance of each of these segments are determined by various analytical algorithms. The top segments are selected on basis of their importance and is output in their temporal order.

However, one problem with this approach is in how the video gets segmented. The segments might have abrupt beginning and ending, so that the viewing experience is compromised. Our video segmentation algorithm subtly avoids this issue. The cuts are placed when there is in fact a change of scene. The *CENTRIST* feature descriptor is good in scene classification, as shown in later experiments.

2.4 Dictionary learning technique

[1] This method does not deal with the concept of superframes or segments. Instead, important frames called *keyframes* are extracted. The method tries to approximate the entire video as a sparse combination of important keyframes. This importance usually translate to the representativeness of the frame. A frame from a group of frames having similar content is extracted and put in the summary. Now the difference between the approximated video, given as a linear combination of the keyframes and the original video is sought to be minimised.

Problem formulation In these methods, the summarisation problem is posed as sparse dictionary learning problem. We seek to find X

$$X^* = \arg\min_X \|Y - DX\|_F^2 + \lambda \|X\|_{2,1}$$

where X is called the pursuit coefficients, D is the dictionary and Y is the given data and λ is the penalization. The subscript F indicates frobenius norm. In this case, the data Y is a matrix whose columns are the feature vectors of frames. The dictionary represents a basis. The features of the given video, can be approximated as the linear combination of the elements in the dictionary. The matrix that does this combination is called the *pursuit coefficient* matrix.

We state the problem of video summarisation as follows. Learn a dictionary D_{key}^* and determine the pursuit coefficients X^* that satisfies

$$D_{key}^*, X^* = \arg\min_{D_key, X} \|D - D_{key}X\|_F^2 + \lambda \|X\|_{1,2}$$

We desire the entries of the pursuit coefficient matrix to be sparse as possible without much affecting the reconstruction of the original video. For this, we have introduced the sparsity constraint. The dictionary matrix will not have the same dimensions of the original data matrix. The number of columns of the dictionary matrix will be way less than that of the original data matrix formed by the vectorisation of the features of frames. The norm that we have used is the $2-1$ norm, that is, take the norm of the rows and then sum them up. This operation will try to make some of the rows of the matrix to be zero. The actual number of zero rows depends on the λ value that we try to enforce. Now, since the λ can become as as large, we redefine the problem to

$$D_{key}^*, X^* = \arg\min_{D_key, X} \frac{\lambda}{2} \|D - D_{key}X\|_F^2 + \frac{1-\lambda}{2} \|X\|_{1,2}$$

which makes the λ value to lie in between 0 and 1. However, this is a non-convex optimization problem.

However, finding a global optimum is not a trivial task for this problem. To overcome this, we will fix the dictionary to represent the original dataset. Therefore, we try to use gradient descend to atleast find a local optimum. Differentiating the above equation, we get the gradient to be

$$(1-\lambda)A^T(AX-A)+\lambda L$$

where

$$L_{i,j} = \frac{\partial}{\partial x_{ij}} \|X\|_{2,1} = x_{ij}\left(\sum_k x_{ik}^2\right)^{-1/2}$$

which we update until convergence. The datapoints are then sorted according to the magnitude of the rows of X. The datapoints corresponding to the largest values of magnitude of rows of X are selected as the keyframes and put in the summary in their temporal order.

2.5 Video segment classification

2.5.1 Neural Network as features

For the experiments with neural networks, we work with identifying important segments within football match videos. The aim is to detect and identify important events like goals, penalty, disciplinary actions which are pertinent to football videos.

2.5.2 YOLO v2

To extract features from football videos, we use the neural network YOLO v2[5], which is a state-of-the-art object recognition network. The network not only predicts the object in the scene , but it also can locate the object within the frame. The system can also detect multiple objects if the objects are of a reasonable size. The network can recognize up to 9000 object classes.

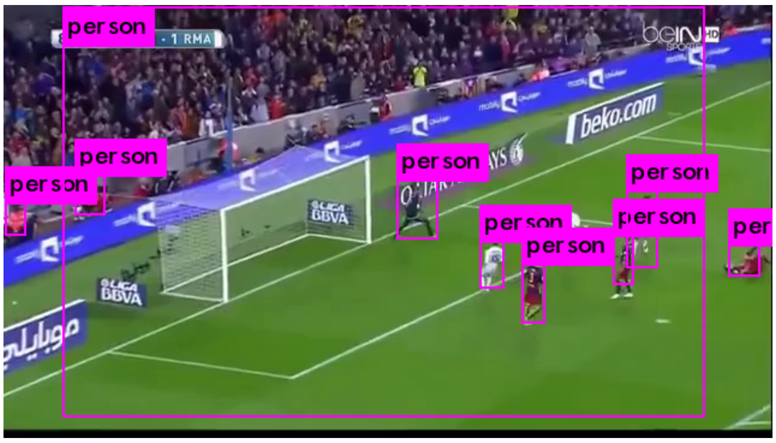

Figure 2.2: The YOLO network can not only predict object in the scene, but it can also given the bounding box location of the object. In our setup , YOLO could process each frame in about 0.7s on a NVIDIA Quadro K20 GPU.

We extract features from the penultimate layer, specifically layer 30 of the darknet neural network. Such features extracted from these layers has a length of 0.15 million floating point numbers. This will be done for each frame of the 20 second video. Therefore, this will result in a huge dimensionality for the input video where the number will be approximately 0.15 million / frame * 20 seconds * 29.97 frames/second = around 90 million floating point numbers, which is too large to process. Therefore, we carry out dimensionality reduction on this data. We construct a histogram on each datapoint on the frame, i.e., we construct 0.15 million histograms that has a bin size of 10. The vectorised form of this histogram becomes our new feature vector for each goal/non-goal example. However, this comes with the problem that this operation destroys the complete temporal information about the given video.

2.6 Auto encoder

An autoencoder is a neural network that can represent data in some compressed format with no supervision. In other words, the autoencoder learns coding for a particular data-item given enough examples. It normally has two components - an encoder and a decoder. The encoder learns some function based on the input data and the decoding part finds out the original input based on the encoder output. This differs from principal component analysis(PCA) in which a linear function is learnt, whereas the autoencoder usually learns a non-linear function. We have used the autoencoder technique to extract features from frames, in replacement to features such as CENTRIST, vggnet and darknet. We have used a convolutional autoencoder for our purpose. We have trained the network using around 6000 images sampled randomly from the shot-detection dataset. The output of the trained encoder was used as an input to our various algorithms of shot detection, segmentation and aummarisation.

2.7 Shot boundary detection

Shot boundary detection is an important part of video summarisation. We can use this information to divide the input video into logical part, and then use selection algorithms to select important segments from these shots. Our experiments were inspired from the paper[2]. In the paper[2], the shot boundaries are detected by training a convolutional neural network. The examples consists of 10 vectorised frames from the videos and the labels for the examples is whether the middle frame is a transition or not. If the central frame is a boundary, we mark the example are positive, otherwise, negative. Now, we normalise the frames by subtracting the average of 10 frames from each of these concatenated frames. So, if there is a transition, the value of the vector tends to be high. However, we try to build a simpler solution to this problem.

The dataset for this problem can be constructed automatically. Pick a video that does not have much transition, select randomly a set of 10 frames. This example, with a high probability will not have any shot boundary in between. Similarly, select 5 frames from one part of the video and select another 5 frames from another part, the timestamps being sampled uniformly. This will create a negative example.

Our model We create the dataset as the paper suggests, but with some minor modifications. Instead of putting the vectorised frames as it is, we extract features like CENTRIST, VGG and autoencoder features. We conduct the experiments for all these features. Each of the datasets contains 1000 positive examples and 1000 negative examples. For the experiments, we implement a simpler solution. Instead of training a neural network, we construct an polynomial kernel SVM and train it with these features. Then we compare the results against each features.

2.8 A naive way to summarise videos

We have tried a new way to summarise videos from the information calculated from the gram matrix. The gram matrix gives us the in-shot variance of any segment (i, j) where i denotes the beginning timestamp/frame and j denotes the ending timestamp or frame. After segmenting the video, we can use a budget, say 10% of the length of the original video. We assume that each of the segments are equally important and needs to be included in the final summary. Therefore, to summarise each segments, we use a sliding window to determine the 10% section of the segment that has the maximum variability. We put these sections from each segment in their temporal order to produce the summary.

2.9 LSTM

Long-Short Term Memory is a kind of neural network that can remember its values over the course of different inputs. Therefore, this kind of network can learn patterns across time. These kinds of neural networks are good at things such as handwriting recognition. Intuitively, they should be good at predicting events such as whether there was a goal or not in a small segment of video. For doing this experiment, we need reasonably small input sizes because the network is memory constrained. Therefore, we carry out PCA to reduce the input data dimensionality. The PCA transform is determined by randomly sampling the dataset and carry out PCA on that. Using the PCA, we transform the input data into smaller dimension and train the output data with LSTM. To use the audio features, we simply concatenate the features obtained from audio with that of video. We repeat the same experiment with LSTM on the newly created dataset.

Chapter 3

Datasets

3.1 Event Dataset

This dataset contains pictures of these sports event categories: rowing (250 images), badminton (200 images), polo (182 images), bocce (137 images), snowboarding (190 images), croquet (236 images), sailing (190 images), and rock climbing (194 images). Images are divided into easy and medium according to the human subject judgement. Information of the distance of the foreground objects is also provided for each image. [3]

3.2 Toy dataset based on MNIST

The MNIST dataset was used to construct a test video for experiments. This dataset contains labeled images of handwritten digits. A video of 1000 frames was constructed by putting one hundred pictures of digit 0, followed by same number pictures of digit 1, until digit 9, totaling to 1000 frames. A wildlife video was used to test the effectiveness of algorithms on general videos.

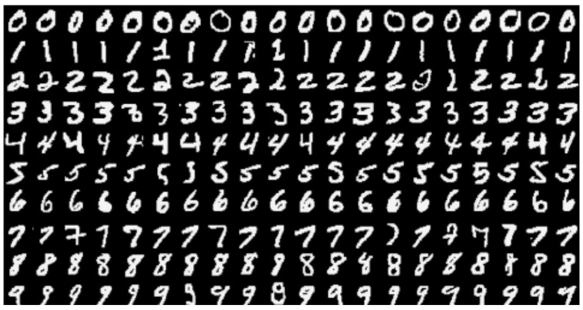

Figure 3.1: The frames of video created from MNIST database

3.3 Football dataset

A goal event video of football dataset was compiled. The dataset contains positive and negative examples of goal events, each of which is of 20 second duration. The positive examples contain a goal event around the center of the video. The negative examples are picked randomly from full match videos, ensuring that it does not contain any valid goal events. Python and bash scripts were used to automate most of the process. The match videos were manually searched for goal events and the timestamp of the goal events with the list of videos were provided to the scripts. The script automatically cuts out the goal events with the required *epilogue* and *prologue* duration. Around 427 negative examples and 351 positive examples were compiled this way.

Chapter 4

Results

4.1 Effectiveness of CENTRIST feature descriptor

4.1.1 Reconstructing the original image from the CENTRIST descriptor

CENTRIST descriptors of small images can be used to reconstruct the original image by minimizing the distance between CENTRIST descriptor of candidate solution and the CENTRIST descriptor of the target solution. The optimization is guided using the simulated annealing algorithm. We have verified the claims that were made in the original paper.

4.1.2 Using the CENTRIST feature descriptor in classification

This experiment was run on the UIUC Sports Event Dataset. Only 3 categories of scenes were selected. The test data and train data were divided randomly into equal size where the entire dataset size was 576 pictures. The SVM library provided by MATLAB was used for classification purpose. The accuracy of classification stood at 76%, verifying the claims made by the paper.

Another experiment was conducted using the MNIST database to recognize the digit *3*. The entire training set and testing set provided by the MNIST database. The testing and training images were transformed into CENTRIST. The CENTRIST transformed training images were then used to fit an SVM. The kernel used was RBF kernel. Then the CENTRIST transformed training images were used to validate the model. The model returned 97.96% accuracy.

4.1.3 Using the CENTRIST feature descriptor to measure similarity between frames

Here we use the centrist feature descriptor to measure the similarity across the frames. A reproduction of similarity matrix for the MNIST video reproduced here. The kernel to measure the similarity of the frames is the Gaussian kernel.

4.2 Effectiveness of the Auto-encoder features

The auto-encoder was used to code and decode some of the pictures drawn from the frames of the dataset. The results are reproduced here. The experiment was carried out with an autoencoder for which there is only the fully connected layer and another with a convolutional setup. The architecture for the auto-encoder is taken from `https://blog.keras.io/`

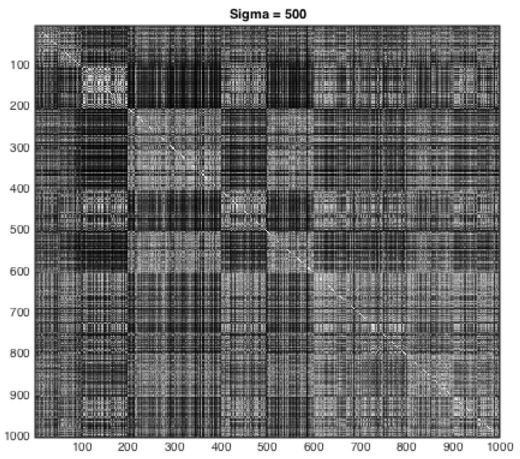

Figure 4.1: The similarity across frames for the MNIST video. The (i, j) pixel denote the similarity of the frame i and the frame j. Lighter colors indicate that the frames are closer.

`building-autoencoders-in-keras.html`. For a quick reference, the architecture is reproduced here.

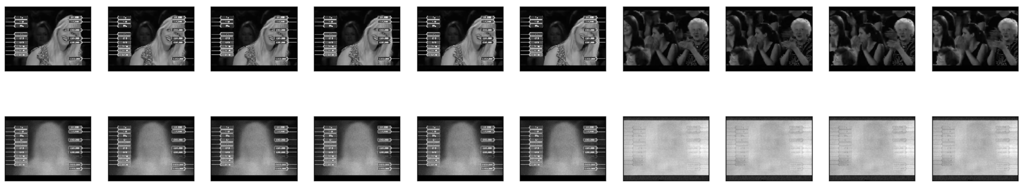

Figure 4.2: The top row shows the original images, while the second row shows the pictures reconstructed by the fully connected network.

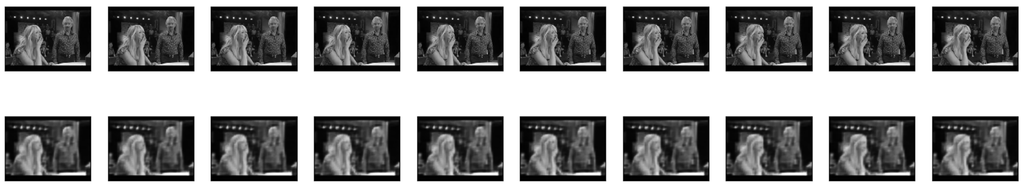

Figure 4.3: The top row shows the original images, while the second row shows the pictures reconstructed by the convolutional network.

Figure 4.4: The autoencoder in action.

4.3 Kernel Temporal Segmentation

The similarity matrix determined above can be used to implement the KTS algorithm. After we determine the dynamic programming table for the number of frames and number of segments, we can choose different segments by just binary searching through the DP table. The program returns a set of timestamps where there is a potential change in scene. We use this information to draw a bar on the video indicating where the cuts have been placed and a slider to indicate the current frame that is being played.

4.4 Shot Boundary Detection

The shot boundary detection experiment was carried out with the following features:

1. CENTRIST feature descriptor
2. VGGnet feature descriptor
3. Autoencoder features
4. Using the frames themselves as the features.

4.4.1 Mahanalobis analysis of the automatically prepared dataset

The multidimensional data normally vary with different axes differently. The Mahanalobis distance is a method that normalises these multiple variations and produces a single scalar value indicating the distance of one data point from the other. Here we analyze the data points produced by the concatenation of CENTRIST feature vector. We draw a cumulative histogram of the negative and positive examples. The distance shown in the graph is the distance between the mean data point of the entire dataset.

4.4.2 CENTRIST

For shot boundary detection, the CENTRIST feature proved very effective. Since the dimension of the descriptors are very small, the size of the training data generated was very small. An analysis of 200 samples each from positive examples, that is to say, those having a cut in between, and negative examples, which lacks a shot boundary is shown.

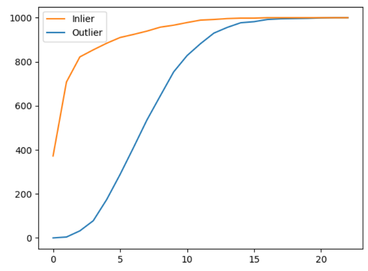

Figure 4.5: The cumulative histogram. If we put a threshold to separate datapoints, the separation could be set to a Mahanalobis distance of 4-5. This indicates that a simple classifier such as SVM should work properly for this task.

Output type	Frequency
Total instances	106
Number of cuts present and reported	93
Number of cuts present, unreported	6
Number of cuts not present, but reported	7

Table 4.1: Results for CENTRIST for shot boundary detection

4.4.3 VGGnet feature descriptor

Compared to CENTRIST, the VGG features proved ineffective.

4.4.4 Autoencoder features

The autoencoder based features failed to produce any results. This might be explained by the gram matrix, which shows no reasonable correlation between the classes.

4.4.5 Using the frames themselves as the features

For this experiment, the frames were converted into grayscale and downsampled so that it could be handled by the SVM. This procedure gave good results.

Output type	Frequency
Total instances	106
Number of cuts present and reported	8
Number of cuts present, unreported	0
Number of cuts not present, but reported	98

Table 4.2: Results for VGGnet for shot boundary detection

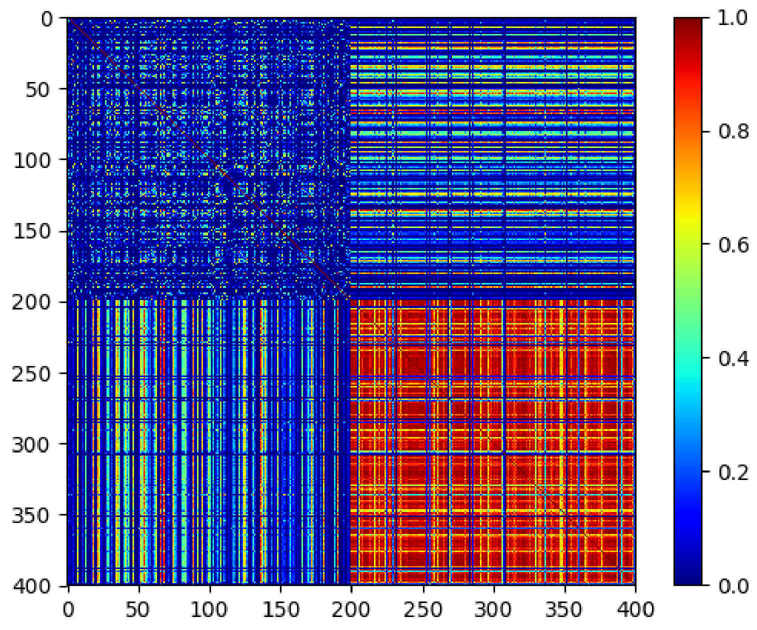

Figure 4.6: The gram matrix for the CENTRIST feature descriptor. The first quadrant is the correlation between positives and negatives, second quadrant shows positives and positives, third shows negatives and positives, the fourth negatives and negatives. The negative examples are clustered.

Output type	Frequency
Total instances	106
Number of cuts present and reported	99
Number of cuts present, unreported	1
Number of cuts not present, but reported	7

Table 4.3: Results for using frames themselves as the features for shot boundary detection

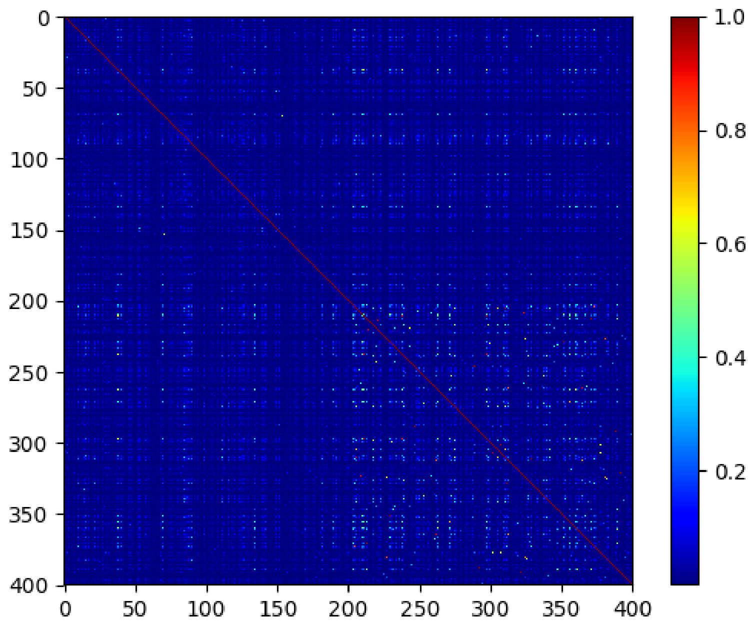

Figure 4.7: The gram matrix for the Autoencoder feature descriptor. The first quadrant is the correlation between positives and negatives, second quadrant shows positives and positives, third shows negatives and positives, the fourth negatives and negatives. The classes has no recognizable correlation.

Demonstration of results A sample video that shows the real transitions and the detected transition is demonstrated. There are two timelines, the top one shows the presence of detected boundaries, and the lower one shows the ground truth.

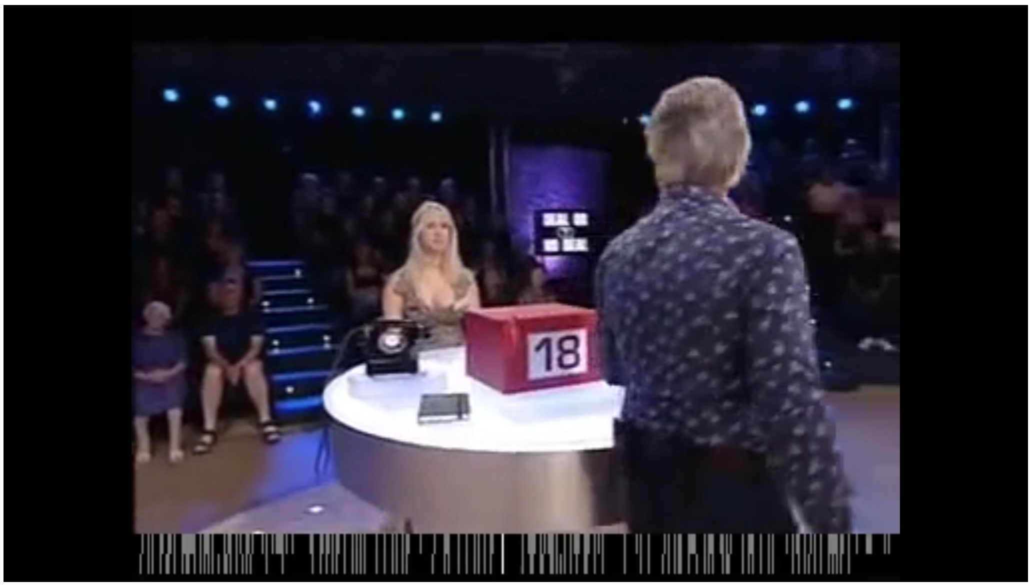

Figure 4.8: Shot detection in action. The timelines are shown in the bottom of the video. The top row shows the predicted shot boundaries, while the bottom row indicates the ground truth. The bright slider indicates the current frame.

4.5 Dictionary based method

The dictionary based method was used on a toy dataset made out the MNIST database. Since there are 10 different kinds of digits in this video, we set the number of clusters to 10. The K-means algorithm was run to identify the clusters and the cluster centres. The points that lie the closest to the mean was designated as a representative point that will represent each of the point within that cluster. This initial set up makes many of the rows of coefficient matrix to be zero, which makes it a reasonable first solution to improve on. The gradient descent algorithm was used to optimize the objective function. However, the objective function just made some small changes to the coefficient matrix and halted. This shows that there are too many local optima. Additional methods should be used to converge the function towards a global optimum. The same experiment was carried out on the wildlife video. It was found that the frames used to cluster around a set of points. A timeline with the selected frames is shown in each experiment.

4.6 Videosummy

The modified Videosummy code was used to generate the summary of some videos. The summary was generated using video and audio features separately. A similar bar has been put on the output video to contrast between the segments of video selected using the fixed size segmenting, KTS and segmentation based on audio features.

4.7 Neural Network as Features

The experiment began by setting up darknet/YOLO v2. This involved setting up darknet on the CUDA platform since the implementation of the darknet is faster on the GPU. After this, the darknet code was modified to take in a set of video frames and extract the output of each layer of the neural net. The penultimate layer was used to extract features and a histogram was used to reduce the dimensionality. The histogram bin size was set to 10.

Experiments for dataset validation A gram matrix was constructed to demonstrate the applicability of the dataset for classification purposes. The distance metrics used were chi-square distance and euclidean distance. It was seen that the matrix showed a high correlation between positive examples while the negative examples were scattered all around the feature space.

The K-nearest neighbour algorithm was used to assess the feasibility of the dataset. We used leave-one-out classification for the dataset. This was done for different values of K ranging from 1 to the minimum number of examples in the either set. Bootstrap sampling was used to create datasets of equal number of positive and negative examples. This experiment was repeated for various dataset sizes of 50, 70, 90, 110, 130. The results are shown here.

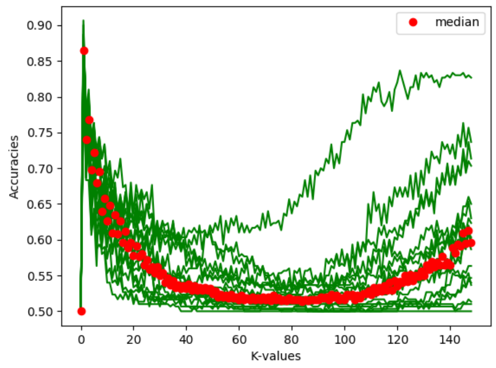

Figure 4.9: The results for validation after bootstrap sampling. There are 20 curves corresponding to leave one out validation of 20 bootstrap samples. The median is shown in the red color.

Classification For classification, a poly SVM model was used to fit the data. For experiments, the training set consisted of equal randomly sampled positive and negative examples of varying sizes. The testing set was prepared using the same method, but with a fixed size of 100 examples. The maximum accuracy for goal/non-goal classification stood at 92%, with 240 positive training examples, 240 negative training examples, test on a randomly sampled data set of 100 positive and 100 negative examples.

4.8 Using the naive summarization method

The naive summarisation method was used to summarise a cartoon video to 20% of its original length. The experiment was conducted using three feature descriptors - CENTRIST, VGGnet

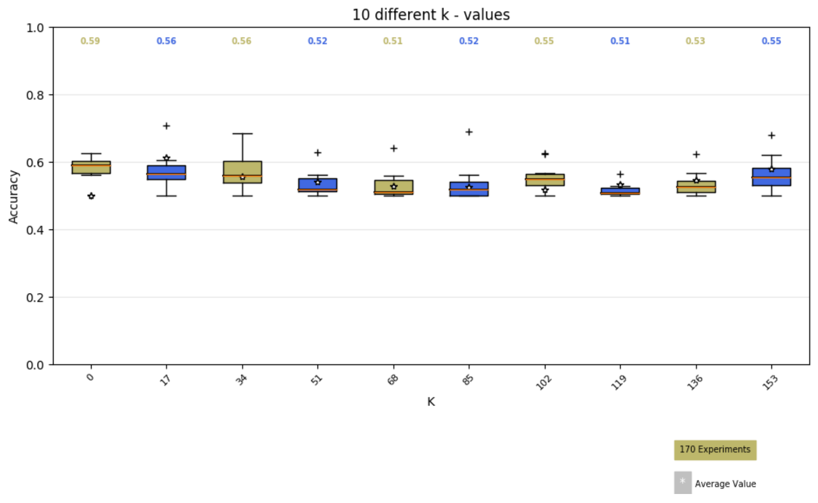

Figure 4.10: A boxplot of accuracies vs K-values for leave-one-out validation for different sample sizes. The accuracies range from 50-60%.

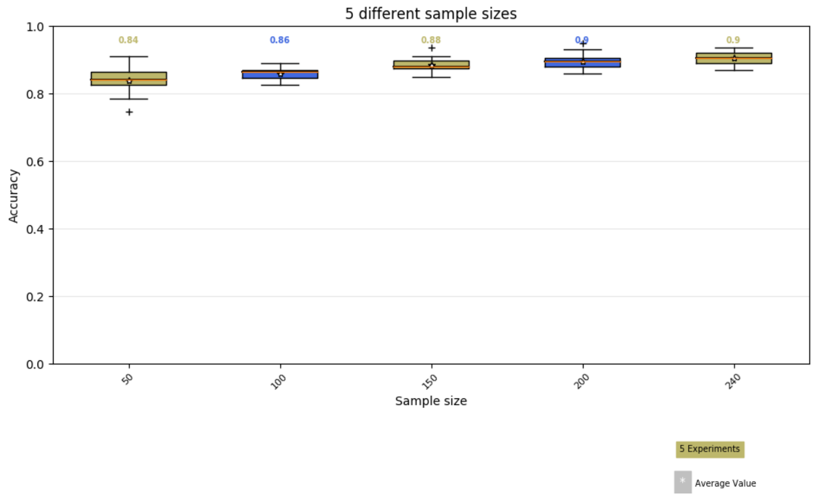

Figure 4.11: A boxplot for the accuracies produced by the poly SVM of order 3. The gamma parameter is $\frac{1}{n}$ where n is the dataset size. The X-axis has results for different sample sizes, 50,100, 150, 200 and 240. The accuracy is found to increase with the increase in training data set.

and Autoencoder features. A storyline of the video and the summarised story is shown below. The text in italics is the summary selected by the corresponding feature descriptor.

CENTRIST Tom attempts to fry the goldfish in the pan. Tom swallows the goldfish. **Jerry takes the pan and smashes it into Tom's face. Jerry opens Tom's mouth like a curtain** and frees the goldfish. **Jerry takes the goldfish in a cup of water and carries it away. Tom follows. Tom gets trapped in a groovy pipe. Tom again follows both.** Tom gets trapped in a hole. **Tom follows Jerry. Jerry uses an iron box to stop Tom.** Tom gets a gun from a wall and shoots at Jerry. **Tom hits the cup in which the goldfish is.** Jerry takes the handle of the broken cup to his hideout without realizing that the goldfish is gone. **Jerry finds the goldfish missing. Tom attempts to fry the goldfish in the fire. Jerry gets Tom trapped in a vessel.** Jerry hits the vessel with a stick to hurt Tom. **Jerry gets the goldfish and takes it away.** They enter a carpet. Tom holds one end of the carpet to send a wave through it. **The wave hits Jerry and goldfish flies back to Tom.** Tom attempts to roast the goldfish in a bread-toaster. **Tom is about to eat the goldfish. Jerry traps Tom in a sheet maker. Tom is made into a sheet. Jerry carries the goldfish away. Tom captures the goldfish using a pan at the entrance of Jerry's house.** Jerry is trapped inside his home using an almarah. Jerry tries to escape but with no success. **Jerry uses the drainage pipes to exit his home. Jerry escapes from his house. Tom cooks the goldfish.** Tom prepares vegetables keeping the goldfish under his feet. **Jerry gets a dynamite and swaps it for Tom's vegetables.** Jerry pulls back the goldfish from Tom's feet and swaps him with Tom's tail. Tom cuts the dynamite and puts it into cooker thinking it to be vegetable. Tom puts his tail into the cooker mistaking it to be the goldfish. **Tom's tail starts to burn. Tom runs for his life with the cooker attached to his tail. Tom traps his tail and the cooker inside the room shutting the door. The cooker explodes. Tom looks at his half burnt tail. Tom is happy.** Tom realizes that he is flying into the outer space due to the explosion. **A still image of an advertisement is shown.**

VGGnet Tom attempts to fry the goldfish in the pan. Tom swallows the goldfish. **Jerry takes the pan and smashes it into Tom's face.** Jerry opens Tom's mouth like a curtain and frees the goldfish. **Jerry takes the goldfish in a cup of water and carries it away.** Tom follows. Tom gets trapped in a groovy pipe. Tom again follows both. Tom gets trapped in a hole. Tom follows Jerry. Jerry uses an iron box to stop Tom. **Tom gets a gun from a wall and shoots at Jerry. Tom hits the cup in which the goldfish is.** Jerry takes the handle of the broken cup to his hideout without realizing that the goldfish is gone. **Jerry finds the goldfish missing.** Tom attempts to fry the goldfish in the fire. Jerry gets Tom trapped in a vessel. **Jerry hits the vessel with a stick to hurt Tom.** Jerry gets the goldfish and takes it away. **They enter a carpet. Tom holds one end of the carpet to send a wave through it.** The wave hits Jerry and goldfish flies back to Tom. Tom attempts to roast the goldfish in a bread-toaster. Tom is about to eat the goldfish. **Jerry traps Tom in a sheet maker.** Tom is made into a sheet. Jerry carries the goldfish away. **Tom captures the goldfish using a pan at the entrance of Jerry's house.** Jerry is trapped inside his home using an almarah. Jerry tries to escape but with no success. **Jerry uses the drainage pipes to exit his home.** Jerry escapes from his house. **Tom cooks the goldfish.** Tom prepares vegetables keeping the goldfish under his feet. **Jerry gets a dynamite and swaps it for Tom's vegetables.** Jerry pulls back the goldfish from Tom's feet and swaps him with Tom's tail. Tom cuts the dynamite

Number of points captured by CENTRIST:	30
Number of points captured by VGGnet:	17
Number of points captured by Autoencoder features	30

Table 4.4: Results for Summarisation algorithm

and puts it into cooker thinking it to be vegetable. Tom puts his tail into the cooker mistaking it to be the goldfish. **Tom's tail starts to burn. Tom runs for his life with the cooker attached to his tail.** Tom traps his tail and the cooker inside the room shutting the door. The cooker explodes. Tom looks at his half burnt tail. **Tom is happy.** Tom realizes that he is flying into the outer space due to the explosion. **A still image of an advertisement is shown.**

Autoencoder Tom attempts to fry the goldfish in the pan. Tom swallows the goldfish. **Jerry takes the pan and smashes it into Tom's face. Jerry opens Tom's mouth like a curtain and frees the goldfish. Jerry takes the goldfish in a cup of water and carries it away. Tom follows.** Tom gets trapped in a groovy pipe. Tom again follows both. **Tom gets trapped in a hole.** Tom follows Jerry. **Jerry uses an iron box to stop Tom.** Tom gets a gun from a wall and shoots at Jerry. **Tom hits the cup in which the goldfish is.** Jerry takes the handle of the broken cup to his hideout without realizing that the goldfish is gone. **Jerry finds the goldfish missing. Tom attempts to fry the goldfish in the fire. Jerry gets Tom trapped in a vessel. Jerry hits the vessel with a stick to hurt Tom.** Jerry gets the goldfish and takes it away. **They enter a carpet.** Tom holds one end of the carpet to send a wave through it. The wave hits Jerry and **goldfish flies back to Tom. Tom attempts to roast the goldfish in a bread-toaster. Tom is about to eat the goldfish. Jerry traps Tom in a sheet maker. Tom is made into a sheet.** Jerry carries the goldfish away. **Tom captures the goldfish using a pan at the entrance of Jerry's house. Jerry is trapped inside his home using an almarah.** Jerry tries to escape but with no success. **Jerry uses the drainage pipes to exit his home.** Jerry escapes from his house. **Tom cooks the goldfish.** Tom prepares vegetables keeping the goldfish under his feet. **Jerry gets a dynamite and swaps it for Tom's vegetables.** Jerry pulls back the goldfish from Tom's feet and swaps him with Tom's tail. **Tom cuts the dynamite and puts it into cooker thinking it to be vegetable. Tom puts his tail into the cooker mistaking it to be the goldfish.** Tom's tail starts to burn. Tom runs for his life with the cooker attached to his tail. **Tom traps his tail and the cooker inside the room shutting the door. The cooker explodes. Tom looks at his half burnt tail.** Tom is happy. **Tom realizes that he is flying into the outer space due to the explosion. A still image of an advertisement is shown.**

Chapter 5

Future Work

Since polynomial kernel SVM gave around 84-93 % success rate for goal detection. This means that with only using visual features, we can identify whether there is a goal or not in any 20 second video segment. This detection technique should be extended to other events such as disciplinary actions against players, goal misses, penalties and the like.
Our other aim is to augment the visual features with audio features to get even more accuracy. Since the visual feature's dimensionality is way too high at 1.5 million numbers per datapoint, as compared to 2560 for audio features, we should try to learn a weight to transform this into a single feature vector, without losing much information. Also, it is required that we look into more datasets or create more datasets on our own, and use the proposed mechanism to identify important events also in that dataset. Some other method should be formulated to identify important event in a video when frames are provided as a stream rather than small segments of video.

We have investigated various methods like video segmentation and shot boundary detection that are first stages of video summarisation. For effective summarisation, we need to include information from the time domain. We need to investigate some methods that can detect more high level features like description of events. If this is possible, then our problem can be reduced to a text-summarisation problem.

Chapter 6

Conclusion

In MTP Stage-I, we have evaluated feature descriptors such as CENTRIST and found it to be good for scene classification. The Kernel Temporal Segmentation algorithm works well for segmentation of videos into subshots. The dictionary based learning methods have poor accuracy owing to the fact that the given objective function is very difficult to optimize. The gradient descent applied on the dictionary based learning methods often converge to a local minima that is not very different from the initial values. The other promising method is to use neural networks as features. Traditional similarity measures between histograms have shown that the positive datapoints are clustered while the negative datapoints are scattered in the feature space. The K-nearest neighbor classification is a weak classifier that provides a little more than 50 per cent accuracy and it should be improvable with the right feature engineering. Until now, the audio features have not been considered in our work. Hopefully, augmenting visual features with audio features might produce a better accuracy of goal detection. However, the visual features are currently of enormous dimension some feature selection should be carried out with dimensionality reduction techniques such as PCA. Future works should include feature engineering by trying out features from different layers of already established neural networks such as darknet/YOLO v2.

In the second phase, we have experimented with different feature descriptors like CENTRIST, VGGnet features and Autoencoder features that were trained by us. We used it for different purposes like segmentation and shot boundary detection. We also proposed a new method for video summarisation using the in-shot variance measure, which indicates diversity in that segment. We have seen that the CENTRIST feature vector performs good for tasks such as segmentation and shot boundary detection. The performance of the summarisation algorithm is difficult to quantify due to its subjective nature.

Bibliography

[1] Pouriya Etezadifar and Hassan Farsi. Scalable video summarization via sparse dictionary learning and selection simultaneously. *Multimedia Tools and Applications*, 76(6):7947–7971, 2017.

[2] Michael Gygli. Ridiculously fast shot boundary detection with fully convolutional neural networks. *arXiv:1705.08214.*

[3] Li-Jia Li and Li Fei-Fei. What, where and who? classifying events by scene and object recognition. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE, 2007.

[4] Danila Potapov, Matthijs Douze, Zaid Harchaoui, and Cordelia Schmid. Category-specific video summarization. In *European conference on computer vision*, pages 540–555. Springer, 2014.

[5] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. *arXiv preprint arXiv:1612.08242*, 2016.

[6] Jianxin Wu and Jim M Rehg. Centrist: A visual descriptor for scene categorization. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1489–1501, 2011.