

# **VIDEO SUMMARISATION**

Enrollment numbers: - 9918103133, 9918103146, 9918103155

Names: - Nishit Anand, Shreya Agarwal, Aditi Dixit

Mentor: - Rupesh Kumar Koshariya



**December - 2021**

**Submitted in partial fulfillment of the Degree of**

**Bachelor of Technology**

**in**

**Computer Science Engineering**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING &  
INFORMATION TECHNOLOGY**

**JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA**

# (I)

## Table of contents

| Chapter No. | Topics   | Page No.  |
|-------------|--|-----------|
| Chapter 1:  | Introduction   |           |
|             | <i>1.1 Abstract</i>  | <i>12</i> |
|             | <i>1.2 Problem Statement</i>                                       | <i>12</i> |
|             | <i>1.3 Significance/Novelty of the problem</i>                     | <i>12</i> |
|             | <i>1.4 Empirical Study</i>   | <i>13</i> |
|             | <i>1.5 Brief Description of the Solution Approach</i>              | <i>13</i> |
|             | <i>1.6 Comparison of existing approaches to the problem framed</i> | <i>13</i> |
| Chapter 2:  | Literature Survey  |           |
|             | <i>2.1 Summary of papers studied</i>                               | <i>14</i> |
|             | <i>2.2 Integrated summary of the literature studied</i>            | <i>31</i> |
| Chapter 3:  | Requirement analysis and solution approach                         |           |
|             | <i>3.1 Overall description of the project</i>                      | <i>32</i> |
|             | <i>3.2 Requirement analysis</i>                                    | <i>32</i> |
|             | <i>3.2.1 Datasets Used</i>   | <i>33</i> |
|             | <i>3.2.2 Technologies Used</i>                                     | <i>34</i> |
|             | <i>3.3 Solution approach</i>                                       | <i>41</i> |
| Chapter 4:  | Modeling and implementation details                                |           |
|             | <i>4.1 Design Diagrams</i>   | <i>46</i> |
|             | <i>4.2 Implementation details</i>                                  | <i>46</i> |
|             | <i>4.3 Risk Analysis and Mitigation</i>                            | <i>51</i> |

|            |                                       |    |
|------------|---------------------------------------|----|
| Chapter 5: | Testing                               |    |
|            | <i>5.1 Testing details</i>            | 54 |
|            | <i>5.2 Issues and limitations</i>     | 55 |
| Chapter 6: | Observations and results              |    |
|            | <i>6.1 Findings</i>                   | 56 |
|            | <i>6.2 Gantt chart</i>                | 57 |
|            | <i>6.3 Future work and conclusion</i> | 58 |
| References |                                       | 59 |

## (II)

### Declaration

We hereby declare that this submission is our own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person nor material which has been accepted for the award of any other degree or diploma of the university or other institute of higher learning, except where due acknowledgment has been made in the text.

Place: Noida

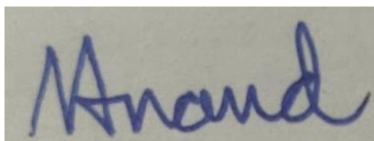
Date: 05-12-2021

Students-

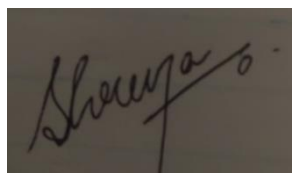
Signature:

Name:

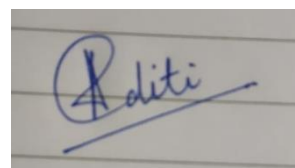
Enrollment No:



Nishit Anand  
(9918103133)



Shreya Agarwal  
(9918103146)



Aditi Dixit  
(9918103155)

**(III)**

**CERTIFICATE**

This is to certify that the work titled “**Video Summarization**” submitted by “**Nishit Anand, Shreya Agarwal , Aditi Dixit** ” in partial fulfillment for the award of degree of Bachelors of Technology of Jaypee Institute of Information Technology, Noida has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor

A handwritten signature in black ink, appearing to read 'Rupesh', is written over a faint, circular official stamp. The signature is slanted and written in a cursive style.

Name: Rupesh Kumar Koshariya

Designation: Assistant Professor (Grade -1)

Date: 05-12-2021

(IV)

**ACKNOWLEDGEMENT**

We deeply acknowledge and extend our thanks to Rupesh Kumar Koshariya sir for his immense support, guidance and constant supervision as well as for providing necessary information regarding the project. We also are grateful for our teachers for providing us with this opportunity to present before them. We would also like to express our gratitude towards our parents and friends for their kind cooperation and encouragement which helped us in completion of this project.

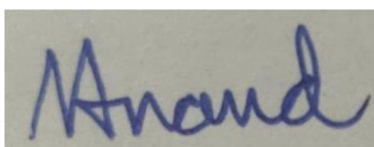
Lastly we would like to extend our sincere thanks to everyone who helped us directly and indirectly in completion of our project.

Sincerely,

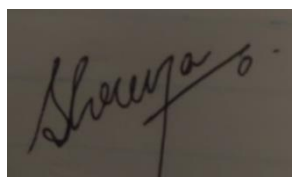
Signature:

Name:

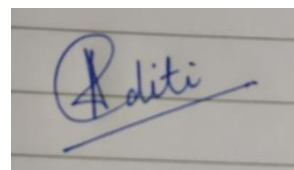
Enrollment No:



Nishit Anand  
(9918103133)



Shreya Agarwal  
(9918103146)



Aditi Dixit  
(9918103155)

Date: 05-12-2021

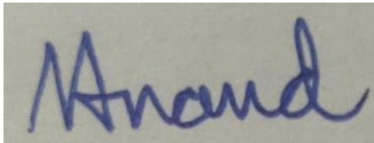
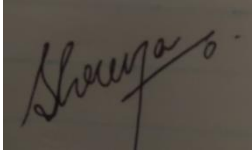
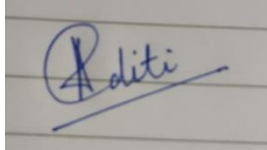
## (V)

### SUMMARY

Video summarization is to generate a short summary of the content of a longer video document by selecting and presenting the most informative or interesting materials for potential users. The output summary is usually composed of a set of keyframes or video clips extracted from the original video with some editing process. The aim of video summarization is to speed up browsing of a large collection of video data, and achieve efficient access and representation of the video content. By watching the summary, users can make quick decisions on the usefulness of the video. Depending on applications and target users, the evaluation of summary often involves usability studies to measure the content informativeness and quality of a summary. For example, in video surveillance, it is tedious and time-consuming for humans to browse through many hours of videos captured by surveillance cameras. If we can provide a short summary video that captures the important information from a long video, it will greatly reduce human efforts required in video surveillance. Video summarization can also provide better user experience in video search, retrieval, and understanding. Since short videos are easier to store and transfer, they can be useful for mobile applications. The summary videos can also help in many downstream video analysis tasks.

---

Signature of Students-

|                |   |  |   |
|----------------|---|--|---|
| Signature:     |  |  |  |
| Name:          | Nishit Anand  | Shreya Agarwal   | Aditi Dixit   |
| Enrollment No: | (9918103133)  | (9918103146)   | (9918103155)  |

Date: 05-12-2021

Signature of Supervisor-



Name: Rupesh Kumar Koshariya

Date: 05-12-2021

**(VI)**

**LIST OF FIGURES**

| <b>Figure</b> | <b>Title</b>  | <b>Page</b> |
|---------------|---|-------------|
| 2.1           | Video summarization   | 14          |
| 2.2           | Unsupervised Video Summarization via Multi-source Features        | 15          |
| 2.3           | Actor-Critic and GAN for Unsupervised Video Summarization         | 17          |
| 2.4           | A Flexible Detect-to-Summarize Network for Video Summarization    | 18          |
| 2.5           | Video Fast-Forwarding via Reinforcement Learning                  | 19          |
| 2.6           | Unsupervised video summarization with deep reinforcement learning | 20          |
| 2.7           | Movie Summarization via Sparse Graph Construction                 | 21          |
| 2.8           | Summarizing Entertainment Video Using Color and Dialogue          | 22          |
| 2.9           | Query-adaptive Video Summarization                                | 23          |
| 2.10          | Video Summarization Using Fully Convolutional Sequence Networks   | 24          |
| 2.11          | Summarizing Videos Unsupervised Attention                         | 25          |
| 2.12          | Unsupervised video summarization framework                        | 27          |
| 2.13(a)       | Visualization of the color histogram features                     | 28          |
| 2.13(b)       | Visualization of the deep features                                | 28          |
| 2.14          | Query-controllable Video Summarization                            | 29          |
| 2.15          | Multi-stream dynamic video Summarization                          | 30          |
| 3.1           | Convolutional Neural Network                                      | 37          |
| 3.2           | Convolutional, Pooling and Fully Connected Layers                 | 38          |
| 3.3           | Long Short Term Memory  | 39          |



|      |   |    |
|------|---|----|
| 3.4  | Google Colab  | 40 |
| 3.5  | Jupyter notebook  | 40 |
| 3.6  | Proposed Approach - Video summarization using deep learning | 41 |
| 3.7  | Extracting Frames from Videos                               | 42 |
| 3.8  | Extracting Features from Frames                             | 43 |
| 3.9  | Convolutional Neural Network                                | 43 |
| 3.10 | Long Short Term Memory                                      | 44 |
| 3.11 | Intersection over Union                                     | 45 |
| 3.12 | Amalgamation of Frames                                      | 45 |
| 4.1  | Design Diagram of our Model                                 | 46 |
| 4.2  | Extracting features from the frames                         | 48 |
| 4.3  | Convolutional Neural Network                                | 48 |
| 4.4  | Dropout Layers  | 48 |
| 4.5  | Long Short Term Memory                                      | 49 |
| 4.6  | Calculation of Temporal Intersection over Union             | 49 |
| 4.7  | Temporal IOU  | 50 |
| 4.8  | Temporal Intersection over Union                            | 51 |
| 4.9  | Combining frames back to make summary video                 | 52 |
| 4.10 | OpenCV  | 52 |
| 4.11 | Training the model on CCTV Surveillance Footage             | 53 |
| 4.12 | Training the model on TVSum and SumMe datasets              | 53 |
| 5.1  | Testing our model on TVSum and SumMe Datasets               | 55 |
| 6.1  | Final F-Scores of our model on TVSum and SumMe Datasets     | 56 |
| 6.2  | Screenshots from CCTV surveillance by our model             | 57 |
| 6.3  | More screenshots from CCTV surveillance                     | 58 |

**(VII)**

**LIST OF TABLES**

| Table | Title   | Page |
|-------|---|------|
| 6.1   | Our Final F-Score as compared to previous work done in this field | 56   |
| 6.2   | Gantt Chart for future plan                                       | 57   |

**(VIII)**

**LIST OF SYMBOLS & ACRONYMS**

| <b>Acronym</b> | <b>Full Form</b>  |
|----------------|---|
| LSTM :         | Long short term memory                                    |
| BiLSTM :       | Bi-long short term memory                                 |
| TVSum :        | Title-based Video Summarization                           |
| CNN:           | Convolutional neural networks                             |
| IOU:           | Intersection over Union                                   |
| DSNet :        | Detect-to-Summarize network                               |
| TF:            | Tensorflow  |
| GAN:           | Generative Adversarial Networks                           |
| ACSUM-GAN:     | Actor Critic Summarisation Generative Adversarial Network |
| FCSN:          | Fully Convolutional Sequence Network                      |
| LTS:           | Long term support   |
| GPU:           | Graphics processing unit                                  |
| CCTV:          | Closed-circuit television                                 |

# Chapter 1 : Introduction

## 1.1 General introduction

With the ever-increasing popularity and decreasing cost of video capture devices, the amount of video data has increased dramatically in the past few years. Video has become one of the most important forms of visual data. Due to the sheer amount of video data, it is unrealistic for humans to watch these videos and identify useful information. It is estimated that it will take around 5 million years for an individual to watch all the videos that are uploaded on the Internet each month in 2021! It is therefore becoming increasingly important to develop computer vision techniques that can enable efficient browsing of the enormous video data. In particular, video summarization has emerged as a promising tool to help cope with the overwhelming amount of video data.

## 1.2 Problem Statement

Given an input video, the goal of video summarization is to create a shorter video that captures the important information of the input video. Video summarization can be useful in many real-world applications. For example, in video surveillance, it is tedious and time-consuming for humans to browse through many hours of videos captured by surveillance cameras. If we can provide a short summary video that captures the important information from a long video, it will greatly reduce human efforts required in video surveillance. Video summarization can also provide better user experience in video search, retrieval, and understanding. Since short videos are easier to store and transfer, they can be useful for mobile applications. The summary videos can also help in many downstream video analysis tasks. For example, it is faster to run any other analysis algorithms (e.g. action recognition) on short videos.

## 1.3 Significance of the problem

The aim of video summarization is to speed up browsing of a large collection of video data and achieve efficient access and representation of video content. By watching the summary users can make quick decisions on the usefulness of the videos. Depending on application and the target audience, the evaluation of summary often involves usability studies to measure the content informativeness and quality of the summary.

## 1.4 Empirical Study

We studied the literature pertaining to the various video summarization techniques by going through the existing research papers. We then tried reproducing the results of a few authors from the above mentioned papers. We analyzed different data sets used for video summarization techniques , the most popularly used being TVSum and SumMe and found the ground truth summaries corresponding to the data sets for training the model. More details about both the datasets have been mentioned further in the report.

As a part of the existing tools survey we studied the unsupervised and supervised methods for video summarization. Also for deep learning we will be using NeuralNetworks and Convolutional Neural Networks. This requires a good amount of computational power and GPU power to do the required ML tasks quickly and efficiently.

## 1.5 Brief description of the solution approach

Our proposed solution is to first divide the video into video frames and then extract features from these frames. These extracted features will be fed to the LSTM network, to determine time-based importance of these frames. Finally they will be classified into a yes or no, whether they will be included in the final summary or not by computing their Temporal Intersection over Union with the ground truth. The data sets used for this approach will be SumMe and TVSum. The detailed overview of the solution approach is mentioned further in the report.

## 1.6 Comparison of existing approaches the problem faced

Most relevant to our approach is the work of online video summarization, which compiles the most salient and informative portion of a video by automatically scanning through the video stream, in an online fashion, to remove repetitive and uninteresting content. Various strategies have been studied, including Gaussian mixture model, online dictionary learning and submodular optimization. Our approach significantly differs from these methods in that it only processes a subset of frames instead of the entire video. To the best of our knowledge, this is the first work to address video summarization by using convolutional neural networks in generating an informative summary from a video.

## Chapter 2: Literature Survey

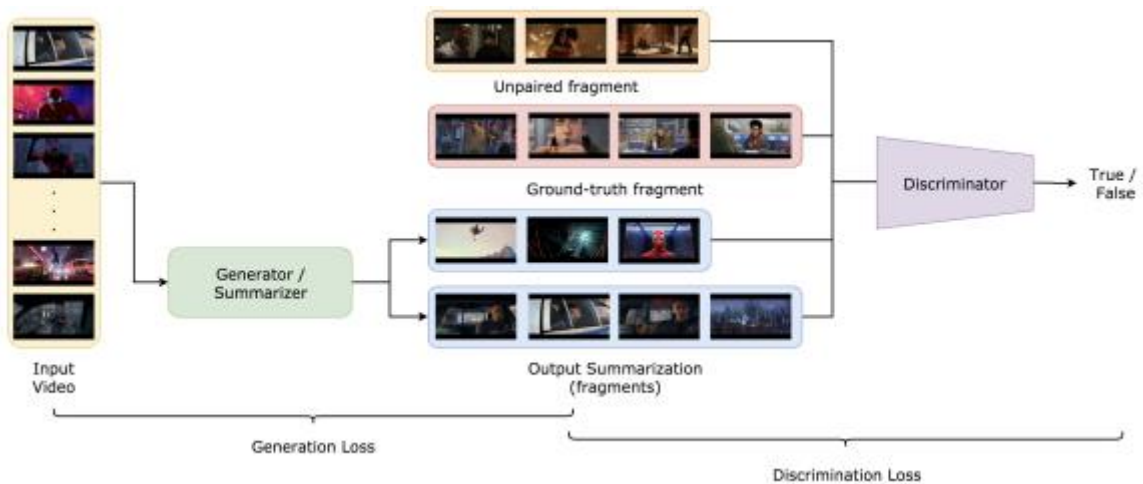
### 2.1 Summary of papers studied

We studied and analyzed the following research papers as a part of a literature survey.

#### 1. Attentive and Adversarial Learning for Video Summarization

By Tsu-Jui Fu, Shao-Heng Tai, Hwann-Tzong Chen

Link to paper: [ResearchPaper 1](#)



(Figure 2.1)-Video summarization

Summary:

This paper aims to address the video summarization problem via attention-aware and adversarial training. The authors formulated the problem as a sequence-to-sequence task, where the input sequence is an original video and the output sequence is its summarization. They proposed a GAN-based training framework, which combines the merits of unsupervised and supervised video summarization approaches. The generator is an attention-aware Ptr-Net that generates the cutting points of summarization fragments. The discriminator is a 3D CNN classifier to judge whether a fragment is from a ground-truth or a generated summarization. The experiments show that the method achieves state-of-the-art results on SumMe, TVSum, YouTube, and LoL datasets with 1.5% to 5.6% improvements.

The Ptr-Net generator can overcome the unbalanced training-test length in the seq2seq problem, and the discriminator is effective in leveraging unpaired summaries to achieve better performance. [1]

Data Sets used: SumMe, TVSum, YouTube, and LoL datasets.

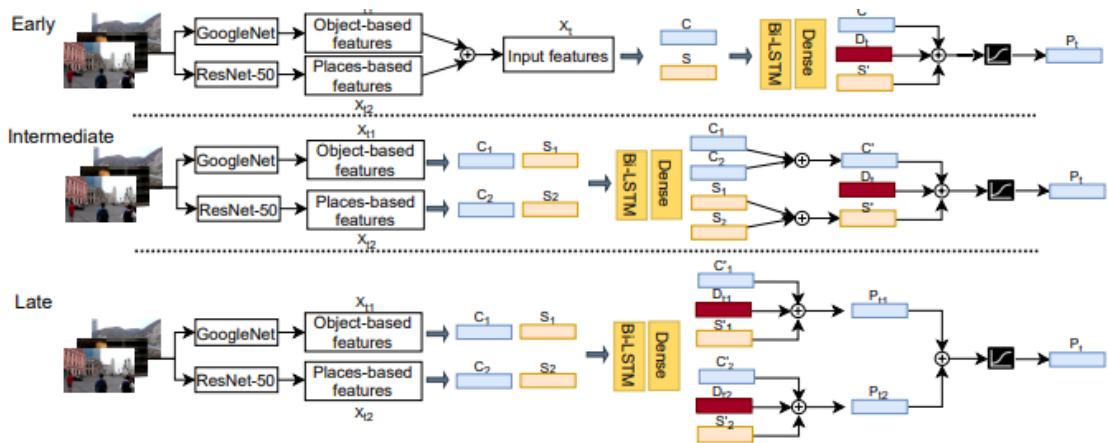
Limitations: During inference, they only performed the generation part, which includes visual feature extraction and ptr-generation, to process the input video. Furthermore, when performing inference, they did not adopt teacher forcing as they have done during training, and therefore the output of predicted summarization fragments may be overlapped or out of order.

Future Scope: . As future work, one can explore more recent semantic segmentation models and develop their counterpart models in video summarization.

## 2. Unsupervised Video Summarization via Multi-source Features

By Hussain Kanafani, Junaid Ahmed Ghauri, SherzodHakimov, Ralph Ewerth

Link to paper: <https://arxiv.org/pdf/2105.12532.pdf>



(Figure 2.2): Unsupervised Video Summarization via Multi-source Features

Summary:

In this paper, the authors proposed a deep learning model for unsupervised video summarization called Multi-Source Chunk and Stride Fusion (MCSF), which investigates the impact of multiple visual representations extracted about visual objects and scene (i.e., places) content. It also uses two temporal constellations of the video features which give the model different perspectives of the video. Consequently, three fusion strategies are suggested and evaluated. For a comprehensive evaluation on the two benchmarks TVSum and SumMe, they compared the method with four state-of-the-art approaches. Two of these approaches were implemented by them to reproduce the reported results. Their evaluation shows that they obtain state-of-the-art results on both datasets, while also highlighting the shortcomings of previous work with regard to the evaluation methodology. Finally, they performed error analysis on videos for the two benchmark datasets to summarize and spot the factors that lead to misclassifications.

DataSets used: SumMe and TVSum. [2]

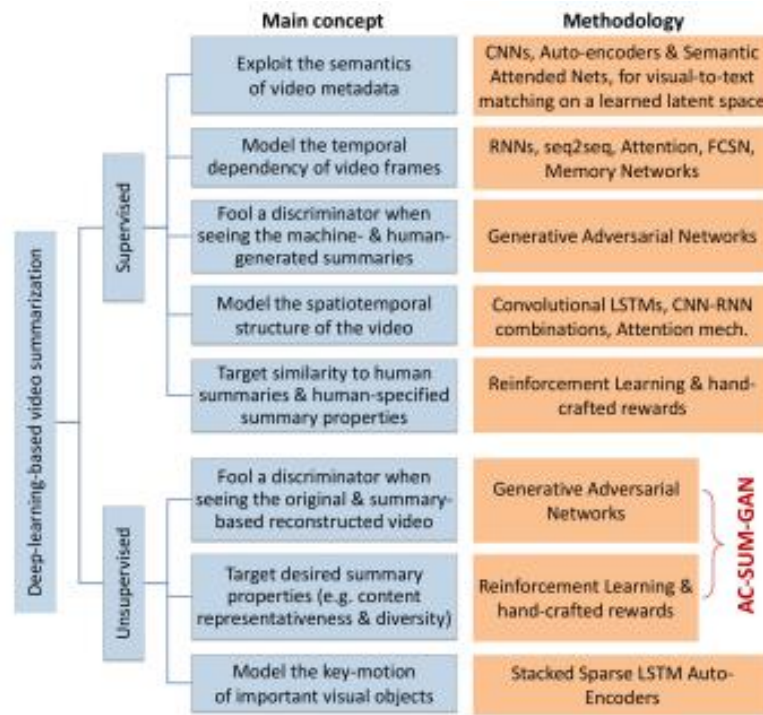
Limitations: Overall, results obtained from unsupervised methods on the original splits were close to the reported ones. Yet, the many videos that were from the test splits (and other videos evaluated twice) led to an unfair evaluation. The error analysis determined that existing methods have difficulties with videos filmed using moving camera settings. These difficulties can be attributed to two main reasons. First, the evaluated methods are basically trained using only object based features that process only frame-level information. Second, those methods create a representative summary that has a similar distribution to the original video without considering the relationships between video segments. Their approach addressed the first issue and presented a corresponding solution but couldn't address the second issue. GAN

3. AC-SUM-GAN: Connecting Actor-Critic and Generative Adversarial Networks for Unsupervised Video Summarization

By E. Apostolidis, E. Adamantidou, A. I. Metsai, V. Mezaris and I. Patras.

Link to paper: <https://doi.org/10.1109/TCSVT.2020.3037883>





(Figure 2.3): Actor-Critic and GAN for Unsupervised Video Summarization

### Summary:

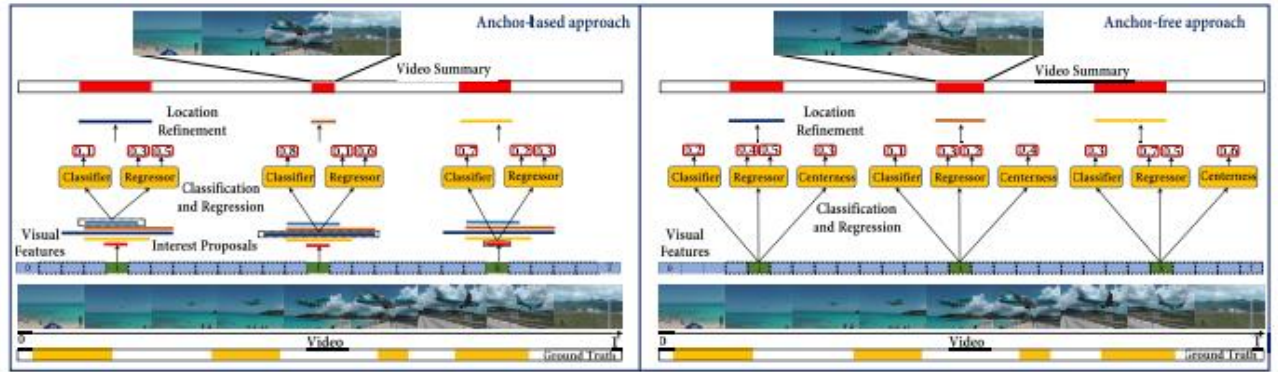
The proposed architecture embeds an Actor-Critic model into a Generative Adversarial Network and formulates the selection of important video fragments (that will be used to form the summary) as a sequence generation task. The Actor and the Critic take part in a game that incrementally leads to the selection of the video key-fragments, and their choices at each step of the game result in a set of rewards from the Discriminator. The designed training workflow allows the Actor and Critic to discover a space of actions and automatically learn a policy for key-fragment selection. Moreover, the introduced criterion for choosing the best model after the training ends, enables the automatic selection of proper values for parameters of the training process that are not learned from the data (such as the regularization factor  $\sigma$ ). Experimental evaluation on two benchmark datasets (SumMe and TVSum) demonstrates that the proposed AC-SUM-GAN model performs consistently well and gives SoA results in comparison to unsupervised methods that are also competitive with respect to supervised methods. [3]

DataSets: SumMe and TVSum

#### 4. DSNet: A Flexible Detect-to-Summarize Network for Video Summarization

By Wencheng Zhu; Jiwen Lu; Jiahao Li; Jie Zhou

Link to paper: <https://ieeexplore.ieee.org/document/9275314>



(Figure 2.4):A Flexible Detect-to-Summarize Network for Video Summarization

Summary:

In this paper the authors have proposed a Detect-to-Summarize network (DSNet) framework for supervised video summarization. The DSNet contains anchor-based and anchor-free counterparts. The anchor-based method generates temporal interest proposals to determine and localize the representative contents of video sequences, while the anchor-free method eliminates the pre-defined temporal proposals and directly predicts the importance scores and segment locations. Different from existing supervised video summarization methods which formulate video summarization as a regression problem without temporal consistency and integrity constraints, their interest detection framework is the first attempt to leverage temporal consistency via the temporal interest detection formulation. [4]

Data sets: TVSum, SumMe, Youtube, Lol.

Limitations: The existing supervised method only learns the importance score of each frame, our anchor-based DSNet approach and formulates video summarization as an interest detection problem and simultaneously learns importance scores and location offsets of generated interest proposals, handling incorrect and incomplete segments.

Future Scope: To eliminate the drawbacks of interest proposals, one can further propose the anchor-free DSNet approach to directly predict the importance scores and segment boundaries. The proposed anchor-based and anchor-free DSNet approaches outperform

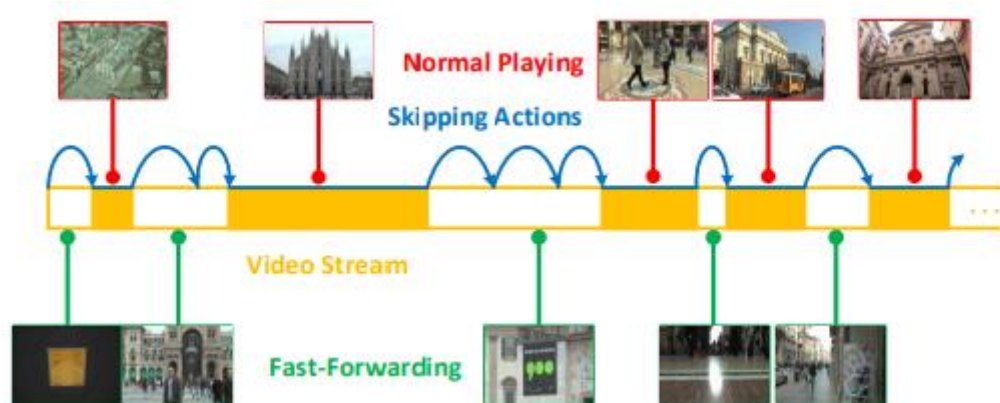
most state-of-the-art supervised methods on the widely-used SumMe and TVSum datasets. In the future, one can attempt to incorporate key shot selection into a unified framework.

## 5. FFNet: Video Fast-Forwarding via Reinforcement Learning

By Shuyue Lan, Rameswar Panda, Qi Zhu, Amit K. Roy-Chowdhury.

Link to paper :

[https://openaccess.thecvf.com/content\\_cvpr\\_2018/papers/Lan\\_FFNet\\_Video\\_Fast-Forwarding\\_CVPR\\_2018\\_paper.pdf](https://openaccess.thecvf.com/content_cvpr_2018/papers/Lan_FFNet_Video_Fast-Forwarding_CVPR_2018_paper.pdf)



(Figure 2.5): Video Fast-Forwarding via Reinforcement Learning

Summary:

In this paper, the authors presented a supervised framework (FFNet) for fast-forwarding videos in an online fashion, by modeling the fast-forwarding operation as a Markov decision process and solving it with a Q-learning method. Quantitative and qualitative results demonstrate that FFNet outperforms multiple baseline methods in both performance and efficiency. It provides an informative subset of video frames that have better coverage of the important content in original video. At the same time, it only processes a small percentage of video frames, which improves computation efficiency and reduces requirements on various resources. In the future, we plan to work on integrating this method with practical system constraints like energy and available bandwidth. [5]

DataSets used: SumMe and TVSum.

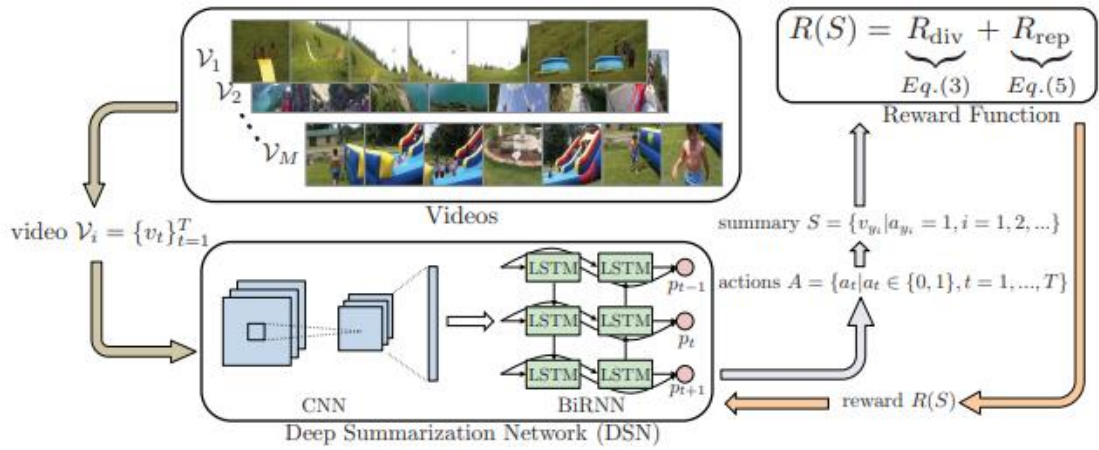
Limitations: On average, it only processes 18.67% of the video frames, which could greatly improve computation efficiency, reduce resource requirement, and lower energy consumption. Note that the requirements on storage and communication are also reduced, but not as much. This is because the neighboring windows of the processed frames are also considered as important for users, and should be stored and transmitted (if needed).

Future scope: It would be interesting to extend our approach by introducing memory in the form of LSTMs—we leave this as part of the future work.

## 6. Unsupervised video summarization with deep reinforcement learning.

By Kaiyang Zhou, Yu Qiao, Tao Xiang.

Link to paper: <https://arxiv.org/abs/1801.00054>



(Figure 2.6):Unsupervised video summarization with deep reinforcement learning.

Summary:

In this paper, the authors formulated video summarization as a sequential decision-making process and developed a deep summarization network (DSN) to summarize videos. DSN predicts for each video frame a probability, which indicates how likely a frame is selected, and then takes actions based on the probability distributions to select frames, forming video summaries. To train their DSN, they proposed an end-to-end, reinforcement learning-based framework, where they designed a novel reward function that jointly accounts for diversity and representativeness of generated summaries and does not rely on labels or user interactions at all. During training, the reward function judges how diverse and representative the generated summaries are, while DSN strives for earning higher rewards

by learning to produce more diverse and more representative summaries. Since labels are not required, our method can be fully unsupervised. Extensive experiments on two benchmark datasets show that our unsupervised method not only outperforms other state-of-the-art unsupervised methods, but also is comparable to or even superior to most of published supervised approaches. [6]

Data Sets: TVSum and SumMe.

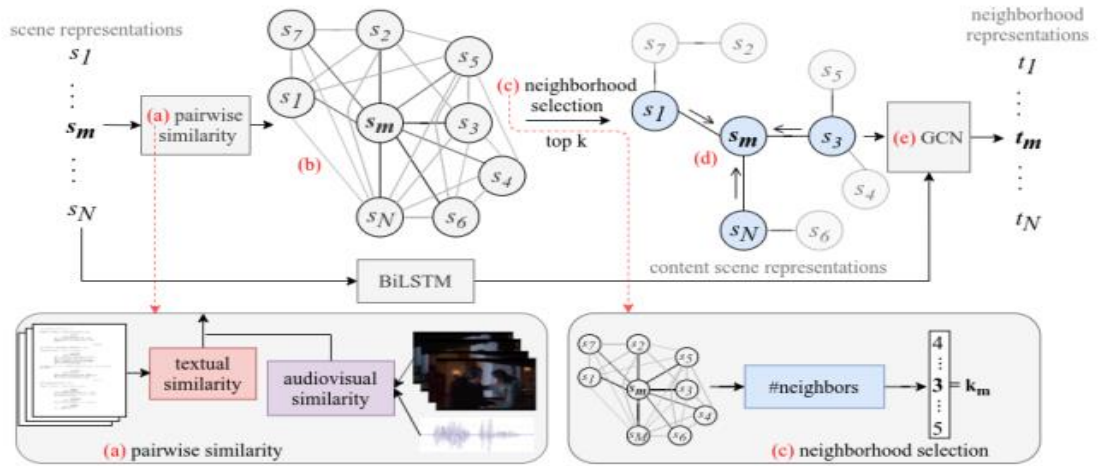
Limitations: The reader should be aware that differences in the variance of the objectives can affect the weights learned. Thus, they should be taken with a grain of salt and only be considered tendencies.

Future Scope: . As future work, one can explore more recent semantic segmentation models and develop their counterpart models in video summarization.

## 7. Movie Summarization via Sparse Graph Construction

By PinelopiPapalampidi Frank Keller Mirella Lapata

Link to paper: <https://arxiv.org/pdf/2012.07536v1.pdf>



(Figure 2.7):Movie Summarization via Sparse Graph Construction

Summary:

In this paper the authors summarized full-length movies by creating shorter videos containing their most informative scenes. They explored the hypothesis that a summary can be created by assembling scenes which are turning points (TPs), i.e., key events in a movie that describe its storyline. They proposed a model that identifies TP scenes by building a



sparse movie graph that represents relations between scenes and is constructed using multimodal information<sup>1</sup>. According to human judges, the summaries created by their approach are more informative and complete, and receive higher ratings, than the outputs of sequence-based models and general-purpose summarization algorithms. The induced graphs are interpretable, displaying different topology for different movie genres. [7]

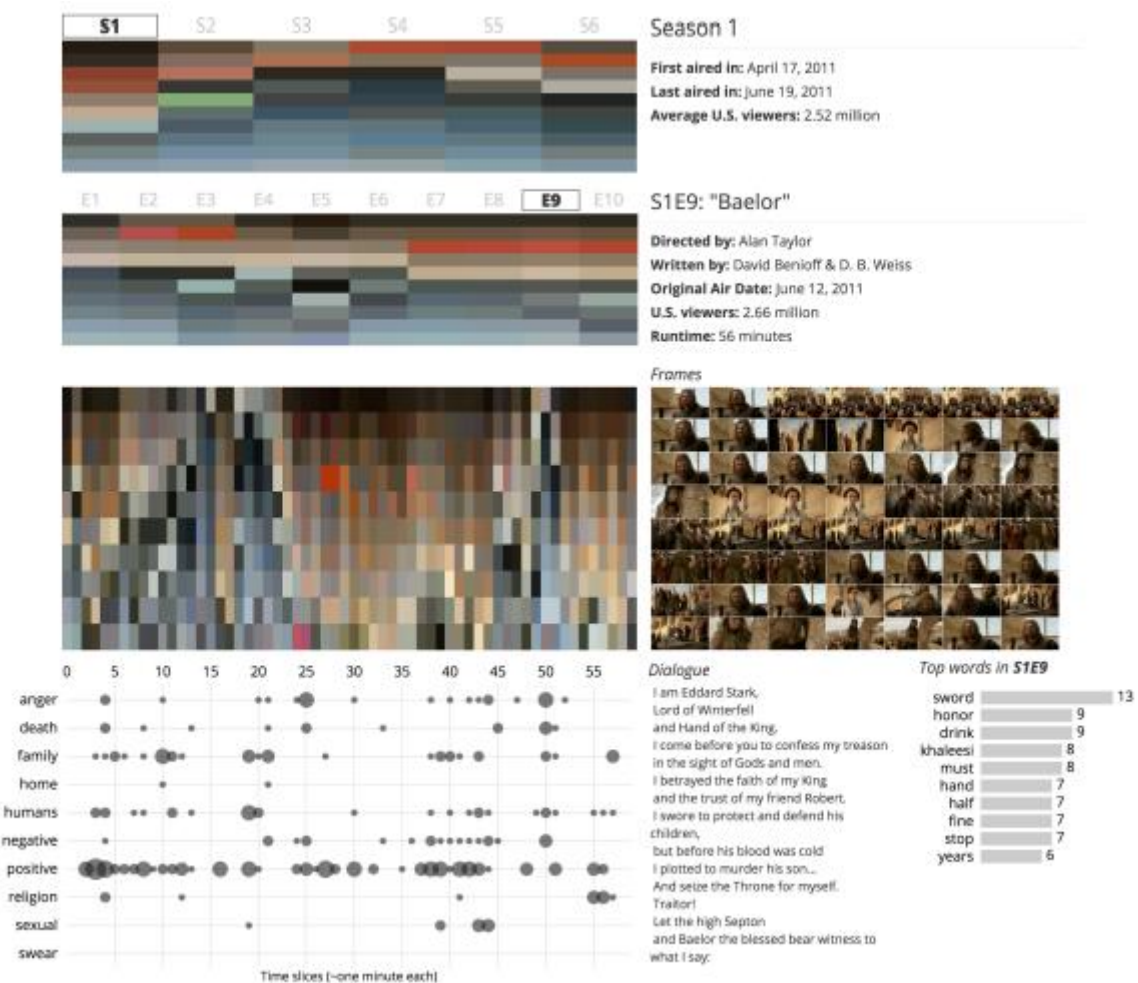
DataSets used: SumMe and TVSum.

Future scope: In the future, one can explore ways to further exploit the graph structure and definitions of TPs in order to produce personalized summaries.

## 8. Summarizing Entertainment Video Using Color and Dialogue

By Fred Hohman, Sandeep Soni, Ian Stewart, John Stasko

Link to the paper: <https://fredhohman.com/a-viz-of-ice-and-fire/>



(Figure 2.8):Summarizing Entertainment Video Using Color and Dialogue

Summary:

Color quantization is a process that reduces the number of distinct colors used in an image, usually with the intention that the new image should be as visually similar as possible to the original image. They used the well-studied median cut quantization clustering algorithm from computer graphics on the data in order to extract the top ten most dominant colors in an image.

Similar to the color extraction, they first segmented each episode into 60 equal-sized time slices and group all dialogue within each slice. We then annotate the dialogue for words that fall into one of the following categories: anger, death, family, home, humans, negative affect, positive affect, religion, swearing, and sexual. [8]

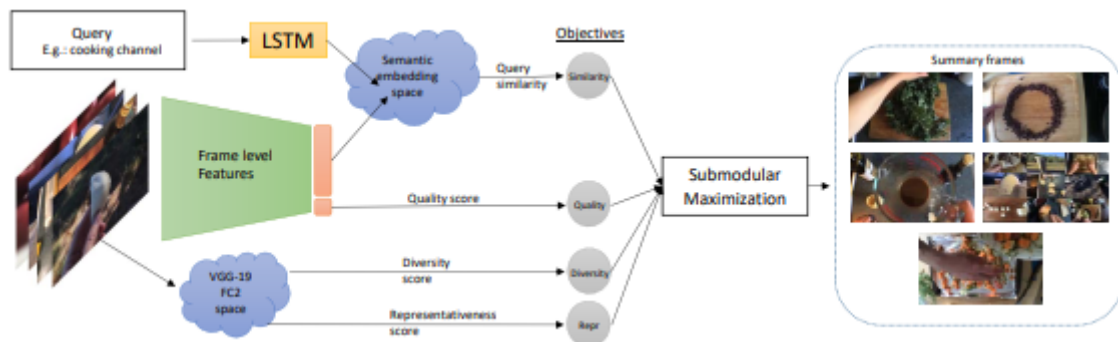
DataSets used: SumMe and TVSum.

Limitations: Because of the limited space, one is only able to show some frames to represent the original video and the corresponding generated video summary in time order.

## 9. Query-adaptive Video Summarization via Quality-aware Relevance Estimation.

By Arun Balajee Vasudevan, Michael Gygli, Anna Volokitin, Luc Van Gool

Link to paper: <https://arxiv.org/pdf/1705.00581.pdf>



(Figure 2.9):Query-adaptive Video Summarization

Summary:

In this paper the authors introduced a new method for query-adaptive video summarization. At its core lies a textual-visual embedding, which let them select frames relevant to a query. In contrast to earlier works, this model allowed them to handle unconstrained queries and even full sentences. They proposed and empirically evaluated different improvements for

learning a relevance model. Their empirical evaluation showed a better training objective, a more sophisticated text model, and explicitly modelling quality leads to significant performance gains. In particular, they showed that quality plays an important role in the absence of high-quality relevance information, such as queries, i.e. when only the title can be used. Finally, they introduced a new dataset for thumbnail selection which comes with query-relevance labels and a grouping of the frames according to visual and semantic similarity. On this data, they tested our full summarization framework and showed that it compares favorably to strong baselines. [9]

DataSets used: SumMe and TVSum.

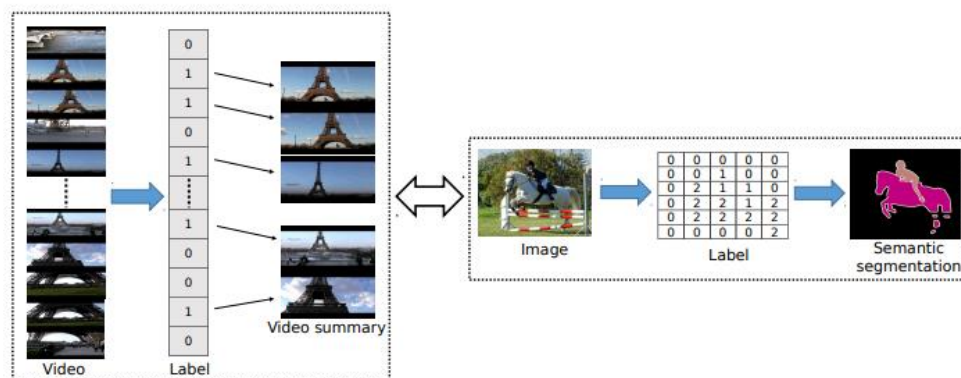
Limitations: The reader should be aware that differences in the variance of the objectives can affect the weights learned. Thus, they should be taken with a grain of salt and only be considered tendencies.

## 10. Video Summarization Using Fully Convolutional Sequence Networks

By MrigankRochan, Linwei Ye and Yang Wang

Link to paper:

[https://openaccess.thecvf.com/content\\_ECCV\\_2018/papers/Mrigank\\_Rochan\\_Video\\_Summ arization\\_Using\\_ECCV\\_2018\\_paper.pdf](https://openaccess.thecvf.com/content_ECCV_2018/papers/Mrigank_Rochan_Video_Summ arization_Using_ECCV_2018_paper.pdf)



(Figure 2.10): Video Summarization Using Fully Convolutional Sequence Networks



Summary:

This paper addresses the problem of video summarization. Given an input video, the goal is to select a subset of the frames to create a summary video that optimally captures the important information of the input video. With the large amount of videos available online, video summarization provides a useful tool that assists video search, retrieval, browsing, etc. In this paper, the authors formulated video summarization as a sequence labeling problem. Unlike existing approaches that use recurrent models, we propose fully convolutional sequence models to solve video summarization. They firstly establish a novel connection between semantic segmentation and video summarization, and then adapt popular semantic segmentation networks for video summarization. Extensive experiments and analysis on two benchmark datasets demonstrate the effectiveness of their models. [10]

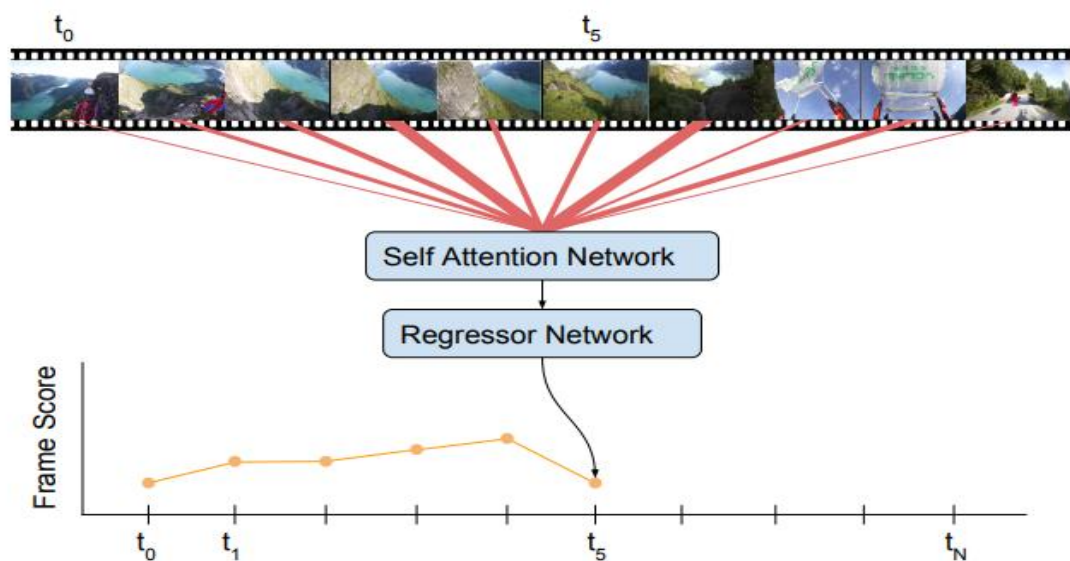
Data Sets: TVSum and SumMe.

Future Scope: . As future work, one can explore more recent semantic segmentation models and develop their counterpart models in video summarization.

## 11. Summarizing Videos with Attention

By Jiri Fajtl, Hajar Sadeghi Sokeh, Vasileios Argyriou, Dorothy Monekosso and Paolo Remagnino

Link to paper: <https://arxiv.org/pdf/1812.01969v2.pdf>



(Figure 2.11): Summarizing Videos with Attention

Summary:

In this work the authors proposed a novel method for supervised, keyshots based video summarization by applying a conceptually simple and computationally efficient soft, self-attention mechanism. Current state of the art methods leverage bi-directional recurrent networks such as BiLSTM combined with attention. These networks are complex to implement and computationally demanding compared to fully connected networks. To that end we propose a simple, self-attention based network for video summarization which performs the entire sequence to sequence transformation in a single feed forward pass and single backward pass during training. Our method sets a new state of the art results on two benchmarks TvSum and SumMe, commonly used in this domain. [11]

Data Sets: TVSum and SumMe.

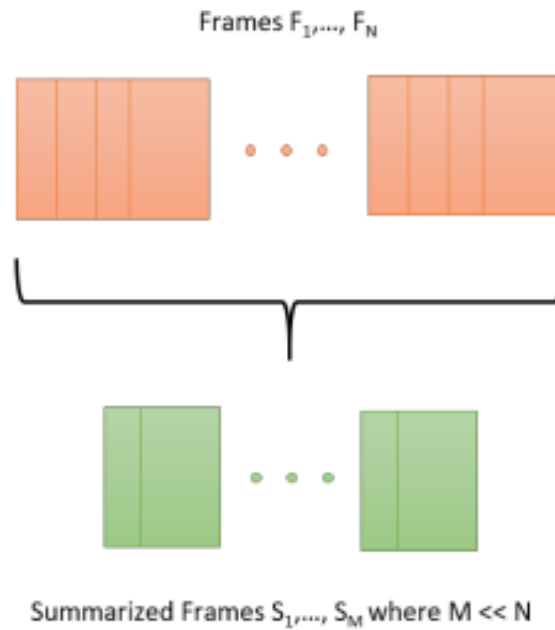
Future Scope: The model is based on a single, global, self-attention layer followed by two, fully connected network layers. The authors intentionally designed and tested the simplest architecture with global attention, and without positional encoding to establish a baseline method for such architectures. Limiting the aperture of the attention to a local region as well as adding the positional encoding are simple modifications that are likely to further improve.

Limitations: Because of the limited space, one is only able to show some frames to represent the original video and the corresponding generated video summary in time order.

12. Unsupervised video summarization framework using keyframe extraction and video skimming.

By Shruti Jadon, Mahmood Jasim.

Link to paper: <https://arxiv.org/pdf/1910.04792v2.pdf>



(Figure 2.12):Unsupervised video summarization framewor

#### Summary:

From what the baseline has been given in the SumMe Dataset, they chose the average human baseline as true, as they would like to consider all perspectives. After testing with all different forms of videos, one can conclude that Gaussian Clustering along with Convolutional Networks can give better performance than other methods with moving point camera videos. In fact, the SIFT algorithm seems to perform well on videos with high motion, the reason behind it is that we used deep layered features, thus they consist of important points inside the image, followed by Gaussian Clustering, which is specifically made for mixture based components. We have also observed that even Uniform Sampling is giving better results for videos which have a stable camera view point and very less motion. One can conclude that one single algorithm can't be the solution of video summarization, it is dependent on the type of video, the motion inside the video. [12]

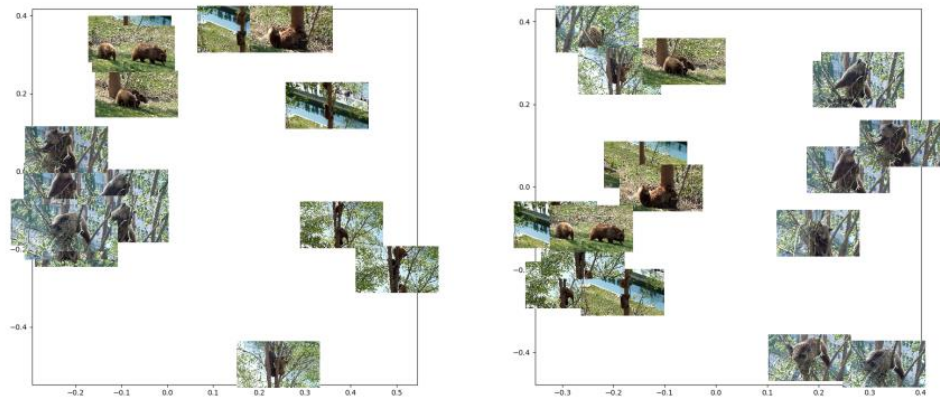
Data Set : SumMe Dataset

Limitations: Video Summarization is one of the hardest tasks because it depends on a person's perception. So, we can never have a good baseline to understand whether our algorithm is working or not. Sometimes, Humans just want 1-2 seconds of video as a summary, whereas machines look for the slightest difference in image intensity and might give us 10 seconds of video.

### 13. ILS-SUMM: Iterated local search for unsupervised video summarization.

By Yair Shemer, Daniel Rotman and Nahum Shimkin.

Link to paper: <https://arxiv.org/pdf/1912.03650v1.pdf>



2.13 (a) Visualization of the color histogram features

2.13(b) Visualization of the deep features

#### Summary:

In this paper, the authors developed ILS-SUMM, a novel video summarization algorithm to solve the subset selection problem under the knapsack constraint. Their algorithm is based on the well-known metaheuristic optimization framework – Iterated Local Search (ILS), known for its ability to avoid weak local minima and obtain a good near-global minimum. Extensive experiments show that their method finds solutions with significantly better total distance than previous methods. Moreover, to indicate the high scalability of ILS-SUMM, they introduced a new dataset consisting of videos of various lengths.[13]

DataSets used: SumMe and TVSum.

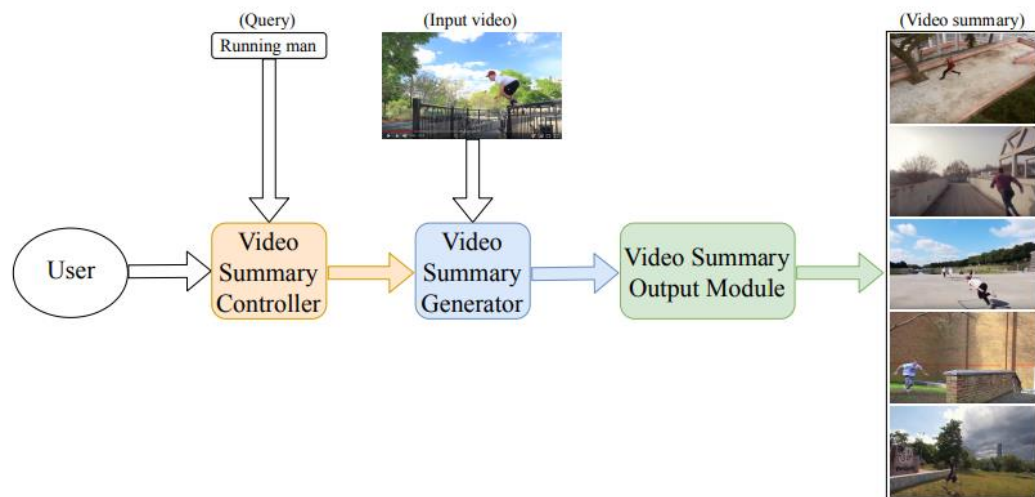
Limitations: Since the question of what is a right evaluation of video summarization is still an open question, there is no solid evidence for an advantage in using deep features rather than color histogram features for this task.

Future Scope: Future research may examine the integration of deep and color histogram features.

## 14. Query-controllable Video Summarization

By Jia-Hong Huang, Marcel Worring

Link to paper: <https://arxiv.org/pdf/2004.03661v1.pdf>



(Figure 2.14):Query-controllable Video Summarization

Summary:

The authors treat a query-controllable video summarization task as a supervised learning problem in this work. To tackle this problem, they proposed an end-to-end deep learning based approach to generate a query-dependent video summary. The proposed method contains a video summary controller, video summary generator, and video summary output module. To foster the query-controllable video summarization research and conduct their experiments, they proposed a new dataset. Each video in the proposed dataset is annotated by frame-based relevance score labels. Their experimental results show that the text-based query not only helps control video summary, but also improves the model performance with +5.83% in the sense of accuracy. [14]

DataSets used: SumMe and TVSum.

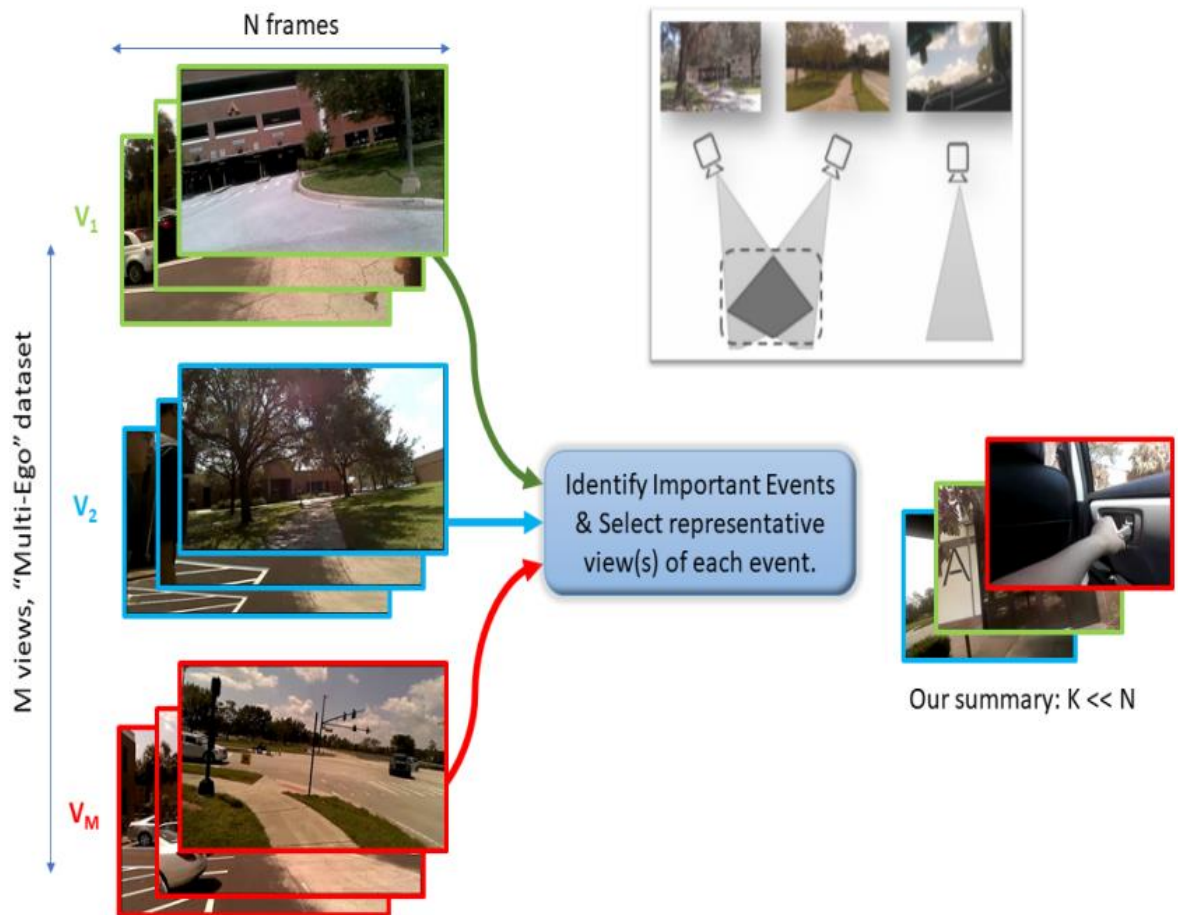
Limitations: Because of the limited space, one is only able to show some frames to represent the original video and the corresponding generated video summary in time order.

Future Scope: Based on the experiment, the authors know that the multi-modal feature fusion method is crucial, so developing a new fusion approach will be interesting future work.

## 15. Multi-stream dynamic video Summarization

By Mohamed Elfeki, Liqiang Wang, and Ali Borji

Link to paper: <https://arxiv.org/pdf/1812.00108v4.pdf>



(Figure 2.15):Multi-stream dynamic video Summarization

Summary:

In this work, the authors proposed the problem of multi-view video summarization for dynamically moving cameras that often do not share the same field-of-view. The formulation provides the first supervised solution to multi-stream summarization in addition to an unsupervised adaptation.

Unlike previous work in multi-view video summarization, they presented a generic approach that can be trained in a supervised or unsupervised setting to generate a

comprehensive summary for all views with no prior assumptions on camera placement nor labels.

It identifies important events across all views and selects the view(s) best illustrating each event. They also introduced a new dataset, recorded in uncontrolled environments including a variety of real-life activities.

When evaluating the approach on the collected benchmark and additional three standard multi-view benchmark datasets, the framework outperformed all baselines of state-of-the-art supervised, reinforcement and unsupervised single- and multi-view summarization methods. [15]

DataSets used: SumMe and TVSum.

Future Scope: . As future work, one can explore more recent semantic segmentation models and develop their counterpart models in video summarization.

## 2.2 Integrated summary of research papers

Given an input video, video summarization aims to produce a shortened version that captures the important information in the video. There are various representations proposed for this problem including video synopsis, time-lapses, montages and storyboards. Early work in video summarization mainly relies on hand-crafted heuristics. Most of these approaches are unsupervised. They define various heuristics to represent the importance or representativeness of the frames and use the importance scores to select representative frames to form the summary video.

Recent work has explored supervised learning approaches for video summarization. These approaches use training data consisting of videos and their ground-truth summaries generated by humans. These supervised learning approaches tend to outperform early work on unsupervised methods, since they can implicitly learn high-level semantic knowledge that is used by humans to generate summaries. Recently deep learning methods are gaining popularity for video summarization.

## Chapter 3: Requirement Analysis and Solution Approach

### 3.1 Overall description of the project

Video summarization is to generate a short summary of the content of a longer video document by selecting and presenting the most informative or interesting materials for potential users. The output summary is usually composed of a set of keyframes or video clips extracted from the original video with some editing process. The aim of video summarization is to speed up browsing of a large collection of video data, and achieve efficient access and representation of the video content. By watching the summary, users can make quick decisions on the usefulness of the video. Depending on applications and target users, the evaluation of summary often involves usability studies to measure the content informativeness and quality of a summary. For example, in video surveillance, it is tedious and time-consuming for humans to browse through many hours of videos captured by surveillance cameras. If we can provide a short summary video that captures the important information from a long video, it will greatly reduce human efforts required in video surveillance. Video summarization can also provide better user experience in video search, retrieval, and understanding. Since short videos are easier to store and transfer, they can be useful for mobile applications. The summary videos can also help in many downstream video analysis tasks.

### 3.2 Requirement Analysis

For Video Summarization, we will be using Deep Learning. And so for that NeuralNetworks and Convolutional Neural Networks will be used. This requires a good amount of computational power and GPU power to do the required tasks quickly and efficiently.

We require the following for the project:-

- A laptop or computer running Windows 10 or macOS Sierra or above or Linux, or Ubuntu 16.04 LTS
- A dual core processor 2 GHz or more
- 8GB of RAM or more
- NVidia GPU 1000 series or above
- Jupyter Notebook
- Google Colab



We also require these things:-

- For training our model we need training data. For this we need video files along with human created ground truth summaries for the video files. We need this, so that the model can learn from it and then be able to identify which frames are important in a video and also for checking Temporal Intersection over Union of the proposed frames with the ground truth frames.
- For validation purposes, we also need testing data, so that we can check

the performance of our model and see how well it can propose summary frames and create a temporally coherent summary.

### 3.2.1 Datasets used

**TvSum:**

## TVSum Dataset

Title-based Video Summarization (TVSum) dataset used in our CVPR 2015 paper "TVSum: Summarizing web videos using titles."



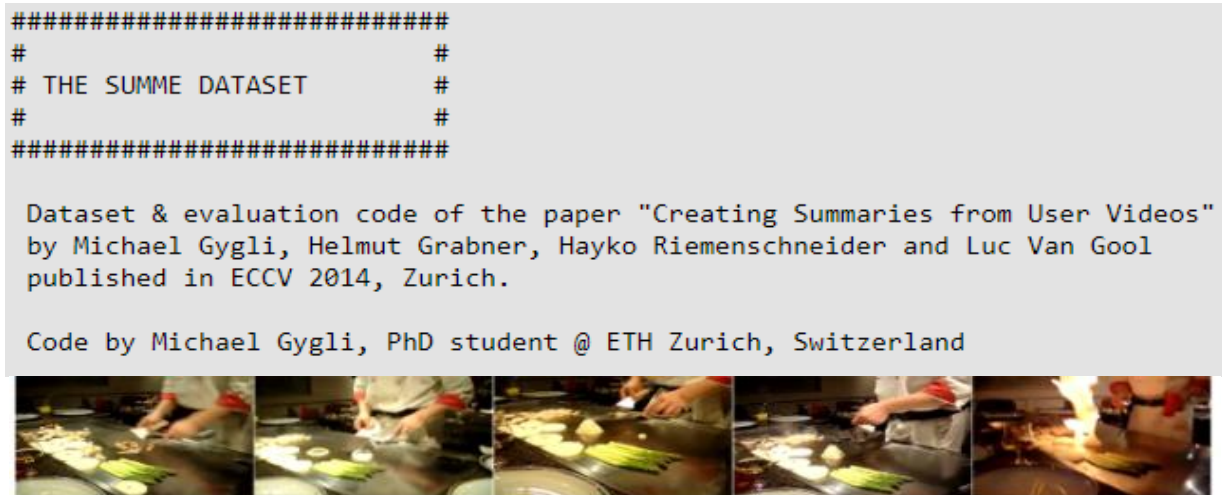
Title-based Video Summarization (TVSum) dataset is used as a benchmark to validate video summarization techniques. It contains 50 videos of various genres (e.g., news, how-to, documentary, vlog, egocentric) and 1,000 annotations of shot-level importance scores obtained via crowdsourcing (20 per video).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

## Contents:

**GrounTruth/** : folder containing the human summary selections

Videos/ : folder with the videos themselves in mp4 (H.264) format

## SumMe:



The SumMe dataset is a video summarization dataset consisting of 25 videos, each annotated with at least 15 human summaries (390 in total).

Contents:

GroundTruth/ : folder containing the human summary selections

Videos/ : folder with the videos themselves in mp4 (H.264) format

### 3.2.2 Technologies Used

#### 1. Programming Language: Python



Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are

available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective.

## 2. Programming Framework

### a. PyTorch :



PyTorch is defined as an open source machine learning library for Python. It is used for applications such as natural language processing. It is initially developed by Facebook artificial-intelligence research group, and Uber's Pyro software for probabilistic programming which is built on it.

Originally, PyTorch was developed by Hugh Perkins as a Python wrapper for the LusJIT based on the Torch framework. There are two PyTorch variants.

PyTorch redesigns and implements Torch in Python while sharing the same core C libraries for the backend code. PyTorch developers tuned this back-end code to run Python efficiently. They also kept the GPU based hardware acceleration as well as the extensibility features that made Lua-based Torch.

Features:

The major features of PyTorch are mentioned below –

Easy Interface – PyTorch offers an easy to use API; hence it is considered to be very simple to operate and runs on Python. The code execution in this framework is quite easy.

Python usage – This library is considered to be Pythonic which smoothly integrates with the Python data science stack. Thus, it can leverage all the services and functionalities offered by the Python environment.

Computational graphs – PyTorch provides an excellent platform which offers dynamic computational graphs. Thus a user can change them during runtime. This is highly useful when a developer has no idea of how much memory is required for creating a neural network model.

The following are the advantages of PyTorch –

- It is easy to debug and understand the code.
- It includes many layers such as Torch.
- It includes a lot of loss functions.
- It can be considered as a NumPy extension to GPUs.
- It allows building networks whose structure is dependent on computation itself.

b. Tensorflow:



TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Important features of TensorFlow –

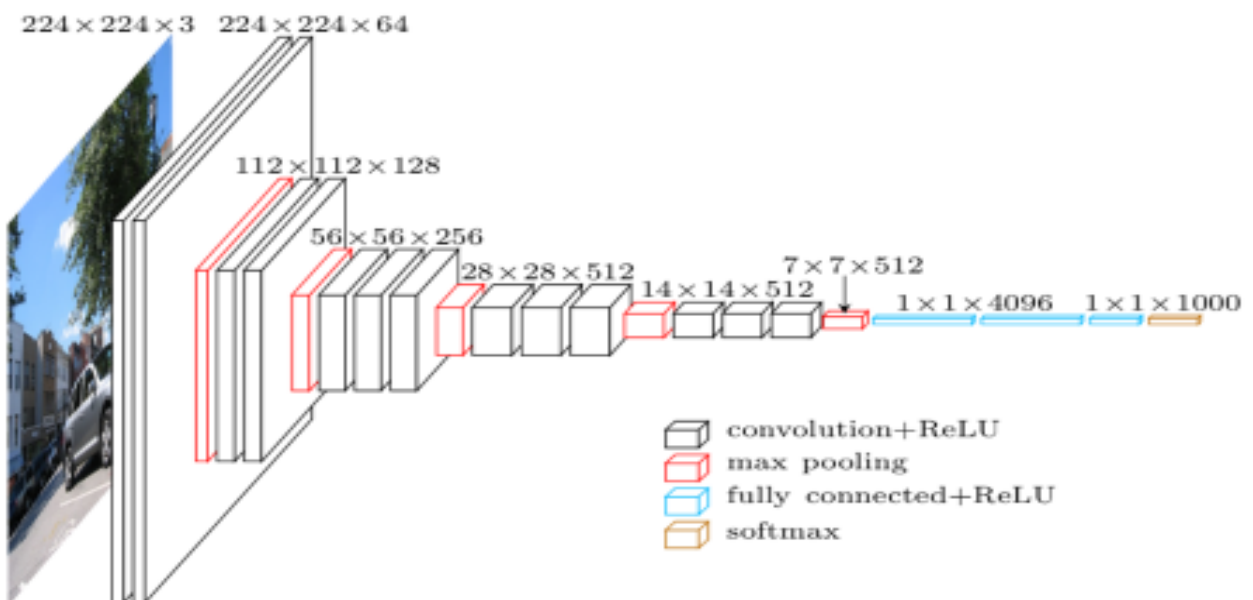
- It includes a feature that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes programming support of deep neural networks and machine learning techniques.
- It includes a highly scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of the same memory and the data used.

TensorFlow is well-documented and includes plenty of machine learning libraries. It offers a few important functionalities and methods for the same.

TensorFlow is also called a “Google” product. It includes a variety of machine learning and deep learning algorithms. TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embedding and creation of various sequence models.

### 3. Neural Networks:

#### a. Convolutional Neural Network (CNN) :



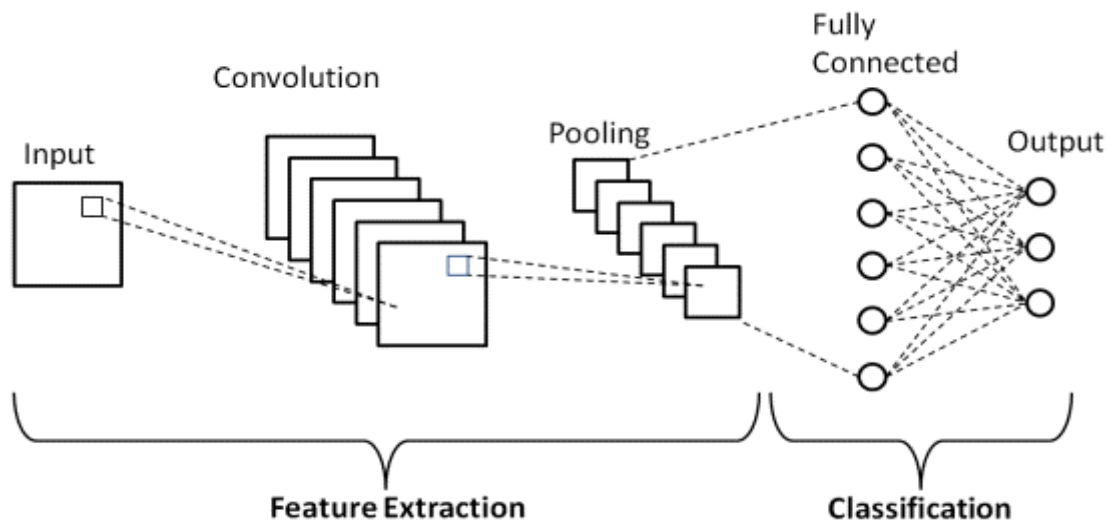
(Figure 3.1) - Convolutional Neural Network

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex.

Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

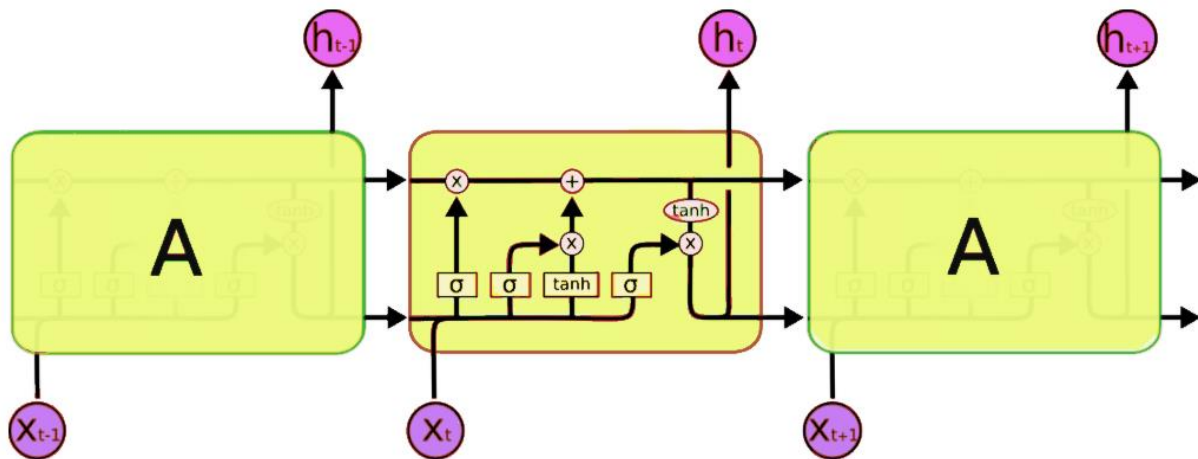


(Figure 3.2) - Convolutional, Pooling and Fully Connected Layers

- Convolutional layer
- Pooling layer
- Fully-connected (FC) layer

The convolutional layer is the first layer of a convolutional network. While convolutional layers can be followed by additional convolutional layers or pooling layers, the fully-connected layer is the final layer. With each layer, the CNN increases in its complexity, identifying greater portions of the image. Earlier layers focus on simple features, such as colors and edges. As the image data progresses through the layers of the CNN, it starts to recognize larger elements or shapes of the object until it finally identifies the intended object.

b. Long Short Term Memory(LSTM) :



(Figure 3.3) - Long Short Term Memory

Long short-term memory (LSTM) [16] networks are a special kind of recurrent neural networks that are capable of selectively remembering patterns for a long duration of time. It is an ideal choice to model sequential data and hence used to learn complex dynamics of human activity. The long-term memory is called the cell state. Due to the recursive nature of the cells, previous information is stored within it. The forget gate placed below the cell state is used to modify the cell states. The forget gate outputs values saying which information to forget by multiplying 0 to a position in the matrix. If the output of the forget gate is 1, the information is kept in the cell. The input gates determine which information should enter the cell states. Finally, the output gate tells which information should be passed on to the next hidden state.

Two of the important variations for the LSTM model are deep LSTM (DLSTM) and CLSTM. DLSTM differs from the general LSTM in the number of layers the model contains. A single-layer LSTM will not be able to obtain well-defined temporal information. However, when more layers are stacked in the LSTM model, it will be able to acquire better temporal features, and hence will be more suitable in capturing motion in the time dimension. In CLSTM, the data are first passed through convolutional layers, which ensure the capturing of spatial features, as shown in Fig. 11.4. The output from the CNN is provided to the LSTM, which will get the temporal features, and hence, the model will capture a motion with respect to both space and time [4]. These two variants can also be combined to give a convolutional deep LSTM, where the outputs from a CNN are given to a multilayer stacked LSTM [16], which is guaranteed to provide a better result at the cost of increased computational complexity.

## 5. IDE

### a. Google Colab :



(Figure 3.4) - Google Colab

Colab is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that you create can be simultaneously edited by your team members - just the way you edit documents in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

As a programmer, one can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

### b. Jupyter Notebook



(Figure 3.5) - Jupyter notebook



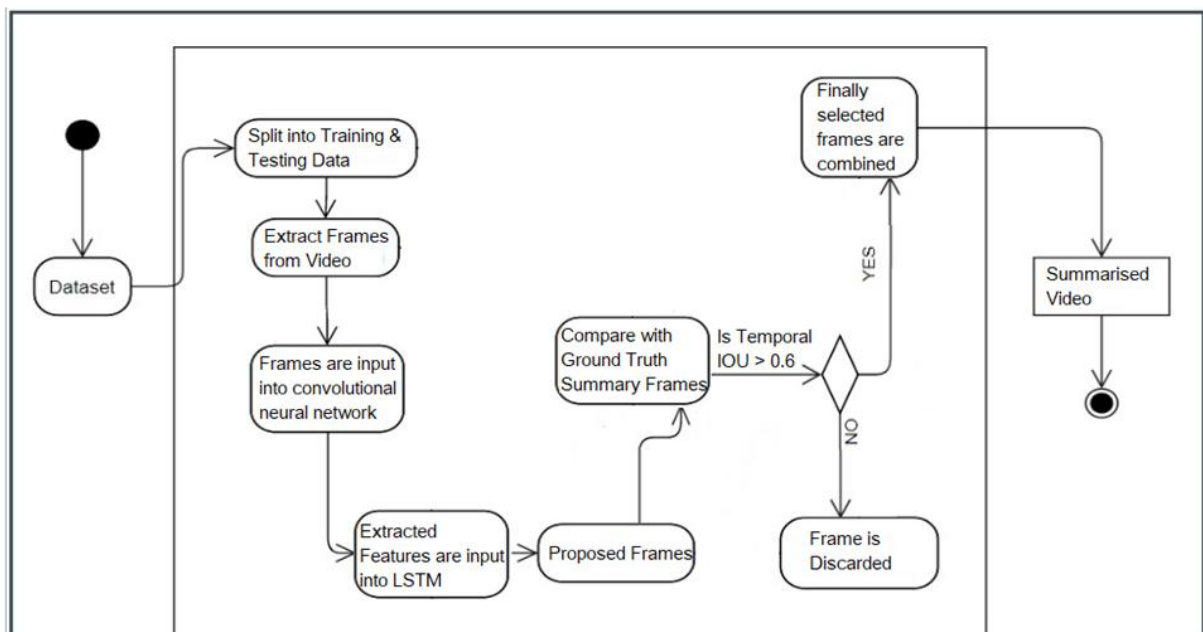
The Jupyter Notebook is an open source web application that you can use to create and share documents that contain live code, equations, visualizations, and text. Jupyter Notebook is maintained by the people at Project Jupyter.

Jupyter Notebooks are a spin-off project from the IPython project, which used to have an IPython Notebook project itself. The name, Jupyter, comes from the core supported programming languages that it supports: Julia, Python,5 and R. Jupyter ships with the IPython kernel, which allows you to write your programs in Python, but there are currently over 100 other kernels that you can also use.

### 3.3 Solution Approach | Proposed Solution

#### Overall Description of Proposed Solution:

Our proposed solution is to first divide the video into video frames and then extract features from these frames. These extracted features will be fed to the LSTM network, to determine time-based importance of these frames. Finally they will be classified into a yes or no, whether they will be included in the final summary or not by computing their Temporal Intersection over Union with the ground truth.



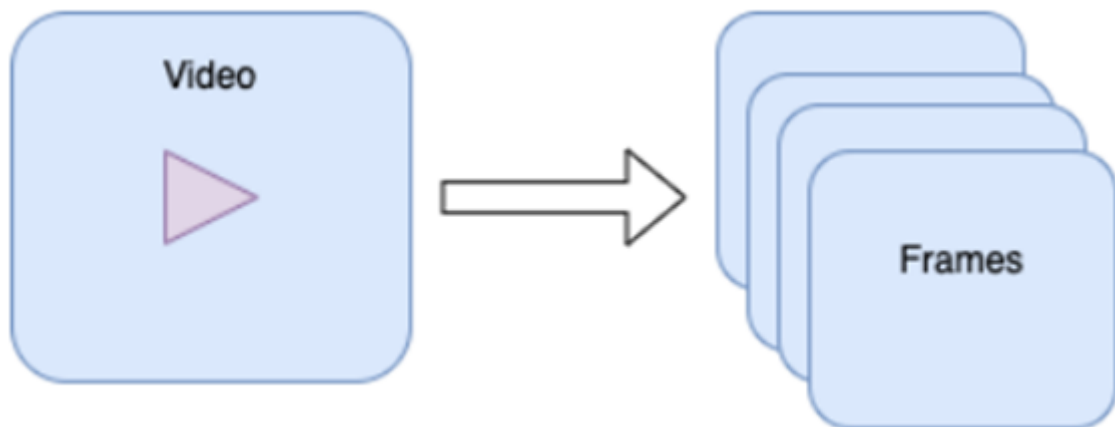
(Figure 3.6) - Proposed Approach - Video summarization using deep learning

Our proposed solution consists of 5 steps:

### 1. **Extracting Frames from video file:**

Any given video consists of a number of video sequences and each video sequence consists of a number of video frames. So we first divide the video into sequences. Then we extract image frames from the video. Thus, we get an image representation of a video in the form of all the frames, which the video is made up of. For this we use the OpenCV library from python.

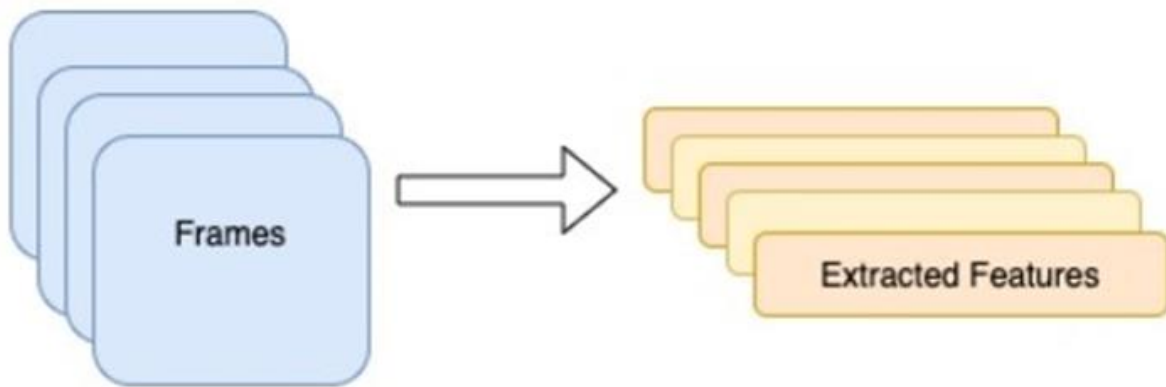
All the videos are 24fps which means that 1 second of video footage consists of 24 image frames. Thus 24 image frames are extracted from every second of video, and are thus put into the model.



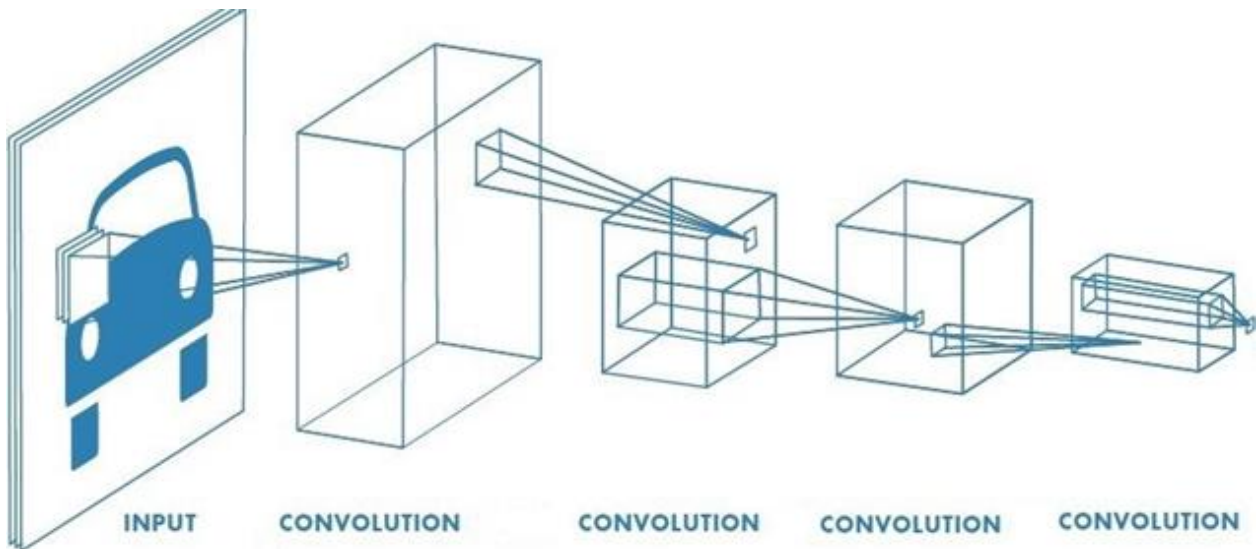
(Figure 3.7) - Extracting Frames from Videos

### 2. **Feature Extraction from the extracted frames:**

Our model needs to learn the features from these extracted frames. For this, we use convolutional neural networks to extract features from the video frames. Now, our model starts learning from the frames extracted from the video in the above step. Moreover, in order to stop overfitting, we have added dropout layers between the convolutional layers. The output of these layers gives us the features our model has extracted from the video frames.



(Figure 3.8) - Extracting Features from Frames



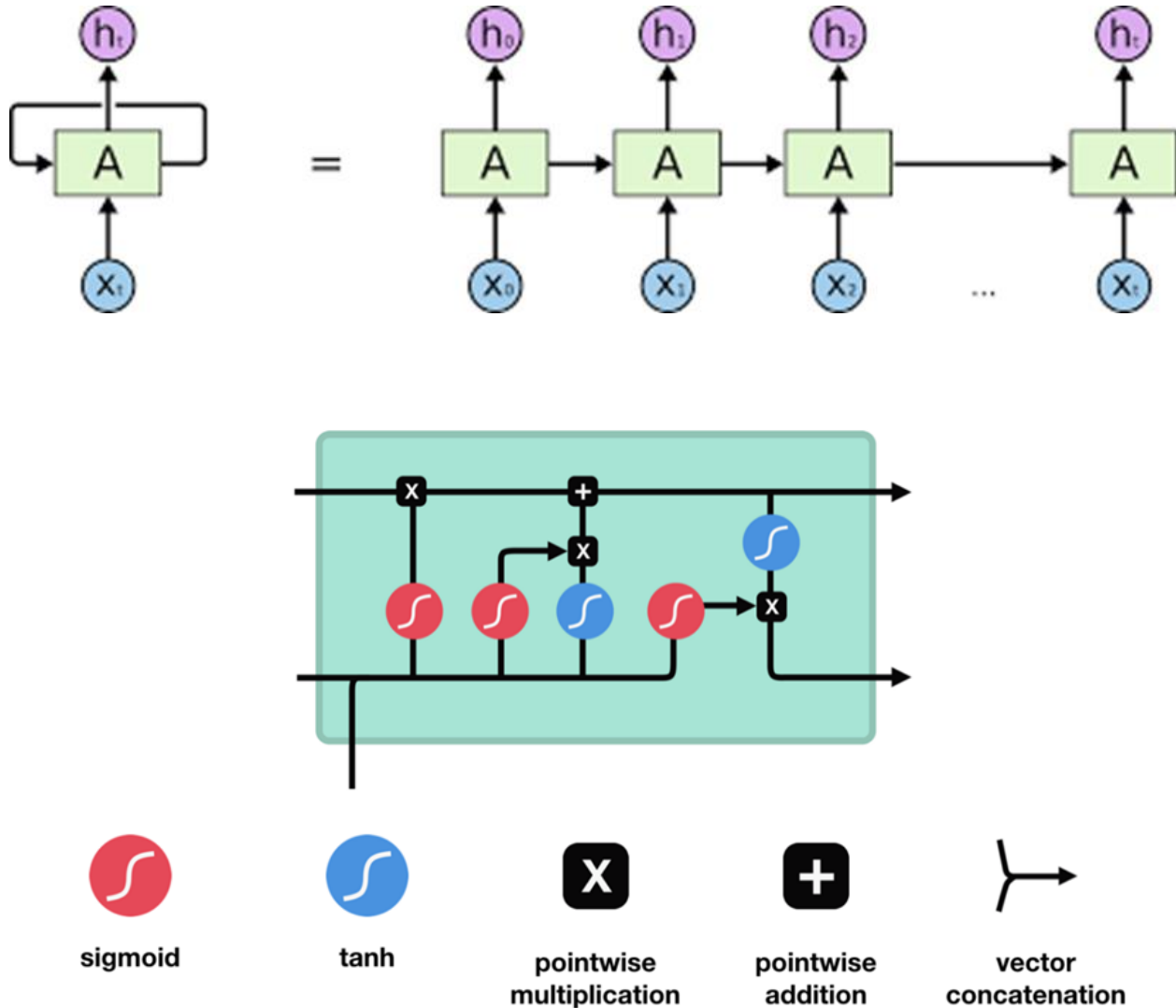
(Figure 3.9) - Convolutional Neural Network

### 3. Temporal interest proposals:

Now we need to determine which video frames are the most important and need to be included in the summary. We use LSTM (Long Short Term Memory) cells in this case. We use LSTM because they are able to pick up information from the previous LSTM cells also. This is important because in a video or movie, time-based interest video segments are very important. In order to determine whether a scene is important or not, we need to know what scene happened before that scene.

Thus, we have used LSTM and so, our model has “attention” and can remember the previous scenes, so it can better determine which scenes have a higher interest value in the

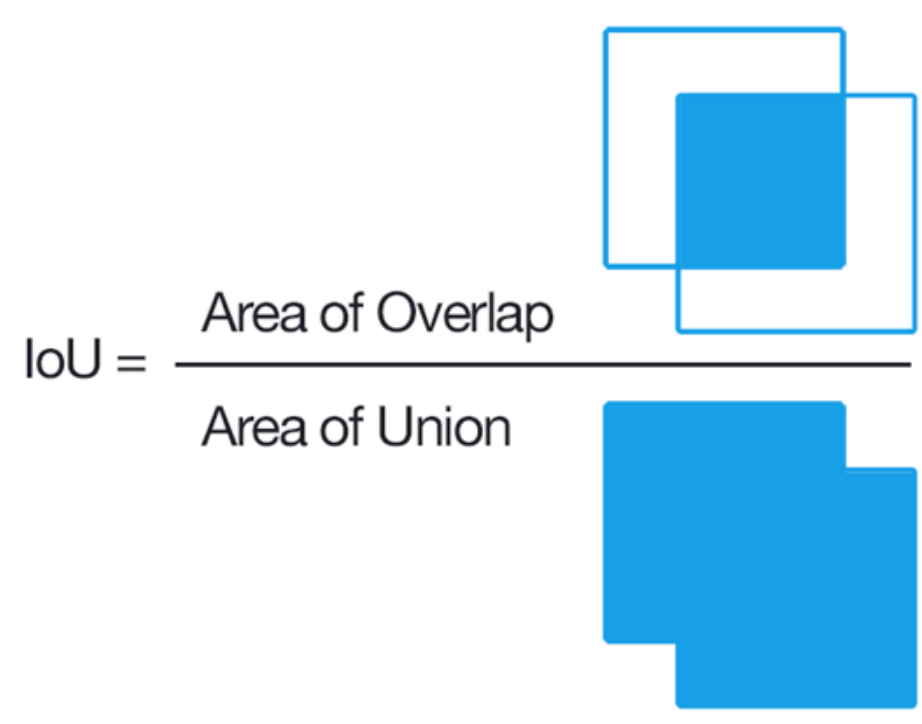
summary. The model then proposes which frames are interesting and important in a time-coherent manner with the help of LSTM.



(Figure 3.10)- Long Short Term Memory

#### 4. Classification on the basis of Temporal Intersection over Union:

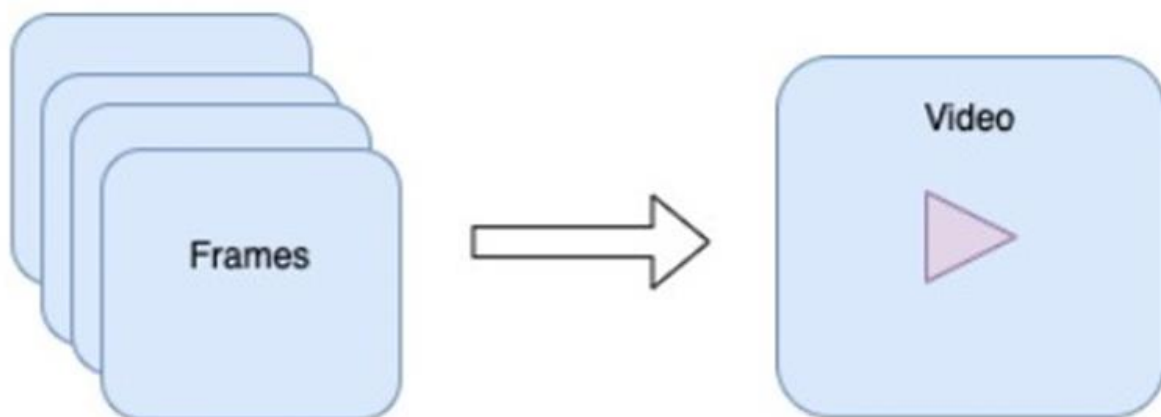
Finally we classify if a frame is going to be put in the final summary or not. The above frames are compared with the Ground Truth summary frames and if their Temporal Intersection over Union (tIOU) is more than 0.6, then we consider that proposal to be positive and is given a score of 1 and thus gets included in the final summary. If the Temporal Intersection over Union is less than 0.6, then that frame is considered negative and is given a score of 0 and thus is not included in the final summary.



(Figure 3.11) - Intersection over Union

## 5. Amalgamation of Frames:

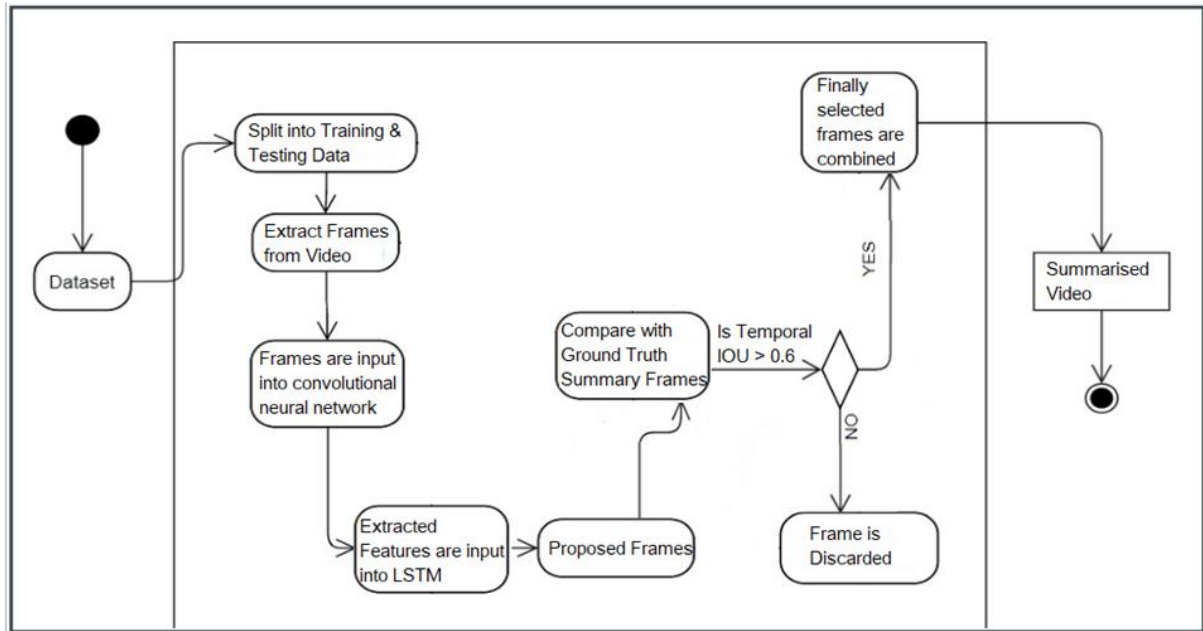
All the frames which are positive and given a score of 1, need to be combined to make the summary video. We use the OpenCV library to combine these frames together and thus the output is the final video summary.



(Figure 3.12) - Amalgamation of Frames

## Chapter 4: Modeling and Implementation Details

### 4.1 Design diagram



(Figure 4.1) - Design Diagram of our Model

### 4.2 Implementation Details

These are the steps we did for implementing our proposed solution:

#### 1. Collecting the datasets:

We collected the TVSum and SumMe Datasets. We sorted them in the correct order i.e. all the video files along with their human created ground truth user summaries. This is important to be done, so that the model can learn and get trained on the input videos.

We also trained the model on CCTV video surveillance footage of length 20 hours for 10,000 iterations. So we collected video data for that also.

## 2. Preprocess the input video file:

We wrote a python program to extract image frames from the video. We used the OpenCV python library for this. This is done, so that the Convolutional Neural Network can learn from these extracted frames



All the videos are 24fps which means that 1 second of video footage consists of 24 image frames. Thus 24 image frames are extracted from every second of video, and are thus put into the model.

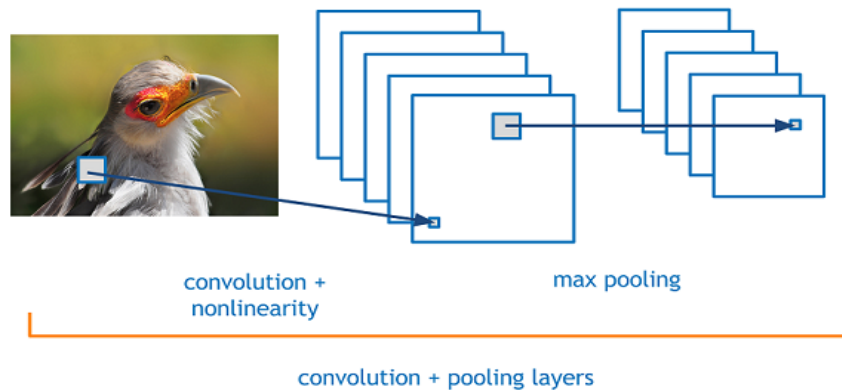
## 3. Detect and Extract Features:

We wrote a custom Machine learning model made up of Convolutional layers and dropout layers in between them. The Convolutional Neural Network extracts spatial features from the images and trains on them, so that it can better predict the right frame to be selected for the summary video.



|                       |               |                            |                                |
|-----------------------|---------------|----------------------------|--------------------------------|
| [2021/09/25 20:18:28] | Epoch: 8/300  | Loss: 0.7023/0.7250/1.4273 | F-score cur/max: 0.6189/0.6285 |
| [2021/09/25 20:18:31] | Epoch: 9/300  | Loss: 0.6874/0.7258/1.4132 | F-score cur/max: 0.6037/0.6285 |
| [2021/09/25 20:18:33] | Epoch: 10/300 | Loss: 0.6691/0.7211/1.3902 | F-score cur/max: 0.6260/0.6285 |
| [2021/09/25 20:18:35] | Epoch: 11/300 | Loss: 0.6643/0.6916/1.3558 | F-score cur/max: 0.6154/0.6285 |
| [2021/09/25 20:18:38] | Epoch: 12/300 | Loss: 0.6523/0.7220/1.3743 | F-score cur/max: 0.6174/0.6285 |
| [2021/09/25 20:18:40] | Epoch: 13/300 | Loss: 0.6333/0.6979/1.3312 | F-score cur/max: 0.6140/0.6285 |
| [2021/09/25 20:18:43] | Epoch: 14/300 | Loss: 0.6321/0.6748/1.3069 | F-score cur/max: 0.6125/0.6285 |
| [2021/09/25 20:18:45] | Epoch: 15/300 | Loss: 0.6221/0.6648/1.2869 | F-score cur/max: 0.6305/0.6305 |
| [2021/09/25 20:18:48] | Epoch: 16/300 | Loss: 0.6118/0.6553/1.2671 | F-score cur/max: 0.6130/0.6305 |
| [2021/09/25 20:18:50] | Epoch: 17/300 | Loss: 0.6026/0.6504/1.2530 | F-score cur/max: 0.6077/0.6305 |
| [2021/09/25 20:18:52] | Epoch: 18/300 | Loss: 0.6001/0.6685/1.2686 | F-score cur/max: 0.6185/0.6305 |
| [2021/09/25 20:18:55] | Epoch: 19/300 | Loss: 0.5809/0.6502/1.2311 | F-score cur/max: 0.6120/0.6305 |
| [2021/09/25 20:18:57] | Epoch: 20/300 | Loss: 0.5701/0.6431/1.2132 | F-score cur/max: 0.6316/0.6316 |
| [2021/09/25 20:19:00] | Epoch: 21/300 | Loss: 0.5741/0.6385/1.2127 | F-score cur/max: 0.6312/0.6316 |
| [2021/09/25 20:19:02] | Epoch: 22/300 | Loss: 0.5608/0.6241/1.1849 | F-score cur/max: 0.6172/0.6316 |
| [2021/09/25 20:19:05] | Epoch: 23/300 | Loss: 0.5562/0.6175/1.1737 | F-score cur/max: 0.6195/0.6316 |
| [2021/09/25 20:19:07] | Epoch: 24/300 | Loss: 0.5497/0.6143/1.1640 | F-score cur/max: 0.6163/0.6316 |
| [2021/09/25 20:19:09] | Epoch: 25/300 | Loss: 0.5456/0.5866/1.1322 | F-score cur/max: 0.6209/0.6316 |
| [2021/09/25 20:19:12] | Epoch: 26/300 | Loss: 0.5324/0.6024/1.1348 | F-score cur/max: 0.6129/0.6316 |
| [2021/09/25 20:19:14] | Epoch: 27/300 | Loss: 0.5167/0.6271/1.1438 | F-score cur/max: 0.6076/0.6316 |
| [2021/09/25 20:19:17] | Epoch: 28/300 | Loss: 0.5318/0.5948/1.1265 | F-score cur/max: 0.6111/0.6316 |
| [2021/09/25 20:19:19] | Epoch: 29/300 | Loss: 0.5243/0.6057/1.1300 | F-score cur/max: 0.6090/0.6316 |
| [2021/09/25 20:19:21] | Epoch: 30/300 | Loss: 0.5178/0.5773/1.0951 | F-score cur/max: 0.6087/0.6316 |
| [2021/09/25 20:19:24] | Epoch: 31/300 | Loss: 0.5020/0.5733/1.0753 | F-score cur/max: 0.6096/0.6316 |
| [2021/09/25 20:19:26] | Epoch: 32/300 | Loss: 0.4968/0.5636/1.0604 | F-score cur/max: 0.6090/0.6316 |
| [2021/09/25 20:19:28] | Epoch: 33/300 | Loss: 0.5022/0.5620/1.0641 | F-score cur/max: 0.5916/0.6316 |
| [2021/09/25 20:19:31] | Epoch: 34/300 | Loss: 0.4909/0.5665/1.0574 | F-score cur/max: 0.6046/0.6316 |
| [2021/09/25 20:19:33] | Epoch: 35/300 | Loss: 0.4924/0.5656/1.0579 | F-score cur/max: 0.6057/0.6316 |

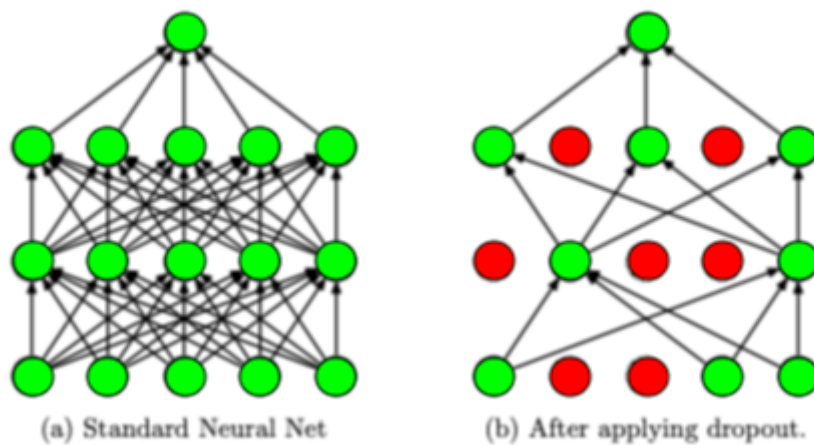
(Figure 4.2) - Extracting features from the frames



(Figure 4.3) - Convolutional Neural Network

#### 4. Reducing Overfitting:

We noticed that the model was overfitting, so we added dropout layers with a chance of 0.5 in between the convolutional layers. This helps reduce overfitting.

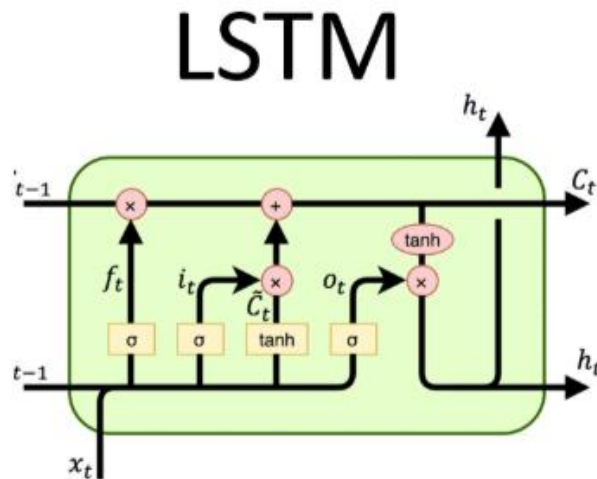


(Figure 4.4) - Dropout Layers



## 5. Feeding these frames into LSTM:

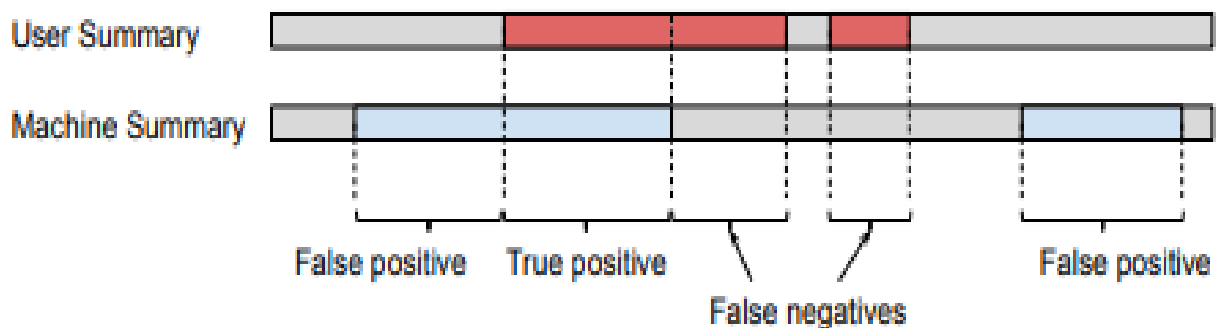
After extracting Spatial Features via Convolutional layers, we have used LSTM to better understand which frames are temporally coherent. As LSTM has “Attention”, it will know which scene occurred before the current scene and will thus be able to determine whether the current frame is important or not. The LSTM layers give output frames which it thinks should be included in the video. These we call as temporal interest proposal frames.



(Figure 4.5) - Long Short Term Memory

## 6. Comparing with ground truth summary video frames:

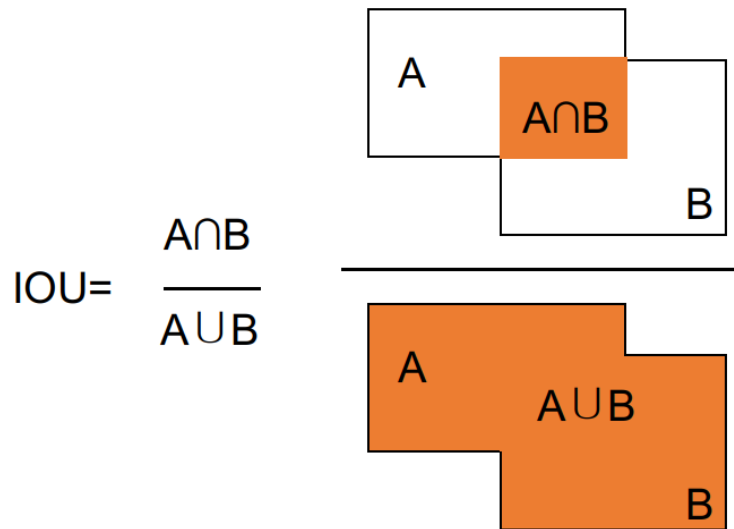
These proposed frames are compared with the frames from the grounds truth summary video, and their Temporal Intersection over union is calculated



(Figure 4.6) - Calculation of Temporal Intersection over Union

## 7. Classifying these frames on the basis of Temporal IOU:

If the Temporal Intersection over Union of a frame is  $>0.6$ , then we label it as positive and it gets included in the final summary video. If the Temporal Intersection over Union of a frame is  $<0.6$ , then we label it as negative and it does not get included in the final summary video.



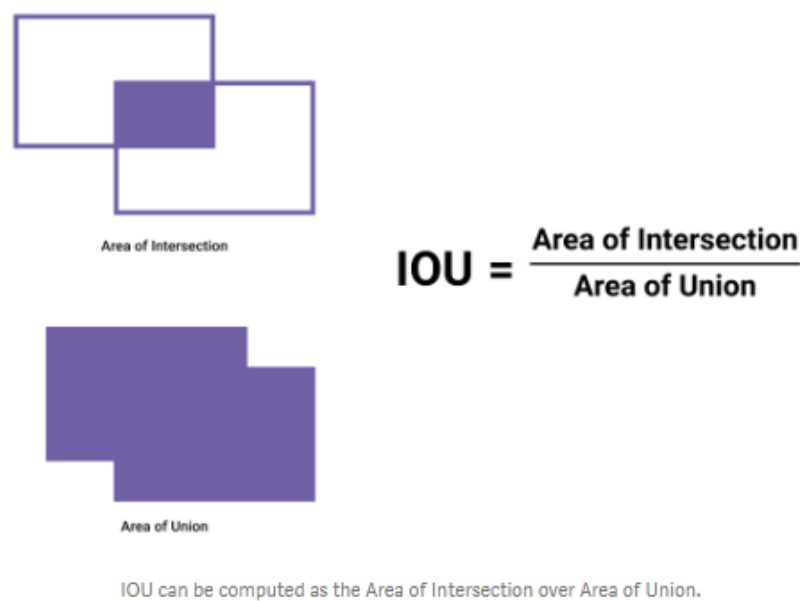
```
nishit@nishit-ubuntu: ~/Downloads/DSNet/src
[2021/10/16 15:04:50] Epoch: 297/300 Loss: 0.3075/0.6518/0.9594 F-score cur/max: 0.9337/0.9337
[2021/10/16 15:04:50] Epoch: 298/300 Loss: 0.3629/0.6322/0.9951 F-score cur/max: 0.9337/0.9337
[2021/10/16 15:04:50] Epoch: 299/300 Loss: 0.3576/0.6959/1.0535 F-score cur/max: 0.9337/0.9337
[2021/10/16 15:04:50] Training done on custom. F-score: 0.8446
nishit@nishit-ubuntu:~/Downloads/DSNet/src$ python evaluate.py anchor-based --model-dir ../models/custom --splits ../custom_data/custom.yml
[2021/10/16 15:04:50] {'model': 'anchor-based', 'device': 'cuda', 'seed': 12345, 'splits': ['../custom_data/custom.yml'], 'max_epoch': 300, 'model_dir': '../models/custom', 'log_file': 'log.txt', 'lr': 5e-05, 'weight_decay': 1e-05, 'lambda_reg': 1.0, 'nms_thresh': 0.5, 'ckpt_path': None, 'sample_rate': 15, 'source': None, 'save_path': None, 'base_model': 'attention', 'num_head': 8, 'num_feature': 1024, 'num_hidden': 128, 'neg_sample_ratio': 2.0, 'incomplete_sample_ratio': 1.0, 'pos_iou_thresh': 0.6, 'neg_iou_thresh': 0.0, 'incomplete_iou_thresh': 0.3, 'anchor_scales': [4, 8, 16, 32], 'lambda_ctr': 1.0, 'cls_loss': 'focal', 'reg_loss': 'soft-iou'}
[2021/10/16 15:04:54] custom split 0: diversity: 0.7782, F-score: 0.9375
[2021/10/16 15:04:54] custom split 1: diversity: 0.5905, F-score: 0.9337
[2021/10/16 15:04:54] custom split 2: diversity: 0.6661, F-score: 0.8745
[2021/10/16 15:04:54] custom split 3: diversity: 0.6432, F-score: 0.5435
[2021/10/16 15:04:54] custom split 4: diversity: 0.5905, F-score: 0.9337
[2021/10/16 15:04:54] custom: diversity: 0.6537, F-score: 0.8446
nishit@nishit-ubuntu:~/Downloads/DSNet/src$
```

(Figure 4.7) - Temporal IOU

## 4.3 Risk Analysis and Mitigation

### 8. Varying Temporal IOU Threshold:

We tested and changed the Temporal IOU Threshold from 0.6 to 0.5 and 0.4, but the summary video obtained in this case was not temporally coherent, and thus produced bad results. We found that the best summarized videos were produced when the Temporal IOU Threshold was set at 0.6

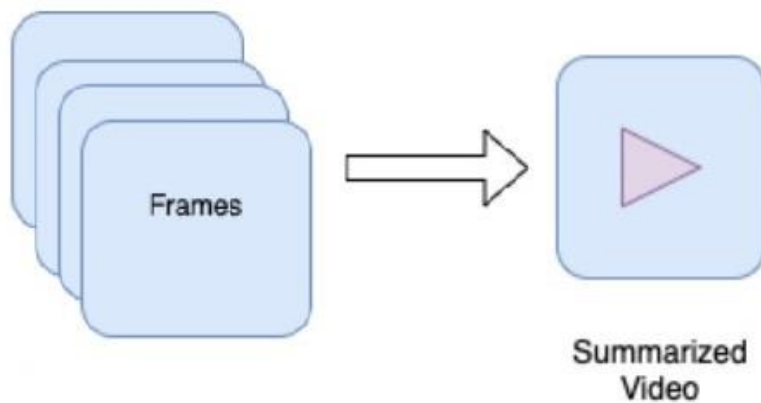


(Figure 4.8) - Temporal Intersection over Union

### 9. Combining the Frames:

After this we combine the frames. We combine the frames at the rate of 24 frames per second, the frame rate at which the videos were originally, so that the output summarized video looks best. We used the OpenCV python library for this.

```
Preprocessing source video ...  
Predicting summary ...  
Writing summary video ...
```



(Figure 4.9) - Combining frames back to make summary video



(Figure 4.10) – OpenCV

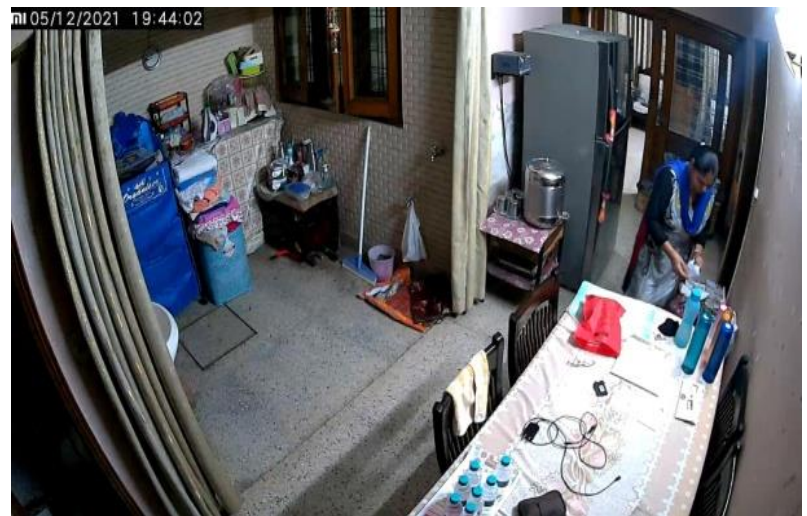
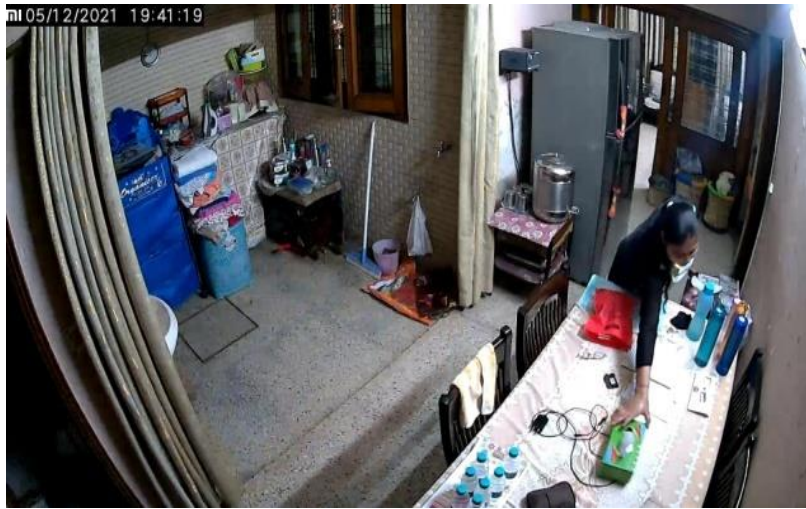
## 10. Training the model:

We trained the model for 10,000 iterations for 30 hours. We made changes to the model multiple times and trained the model again to increase its accuracy.

Next, we trained the model on CCTV video surveillance footage of length 20 hours for 10,000 more iterations. This was done in order to make the model more generalised and so that it is better able to learn the important segments in a CCTV video clip.

After this the model became robust and was able to find which parts of the CCTV video footage have humans and the model was able to separate them out pretty well from the rest of the clip and able to create a good summary video containing all the parts having human activity.





(Figure 4.11) - Training the model on CCTV Surveillance Footage

```
nishit@nishit-ubuntu: ~/Downloads/DSNet/src
2021/09/25 20:59:50] Epoch: 286/300 Loss: 0.1905/0.2707/0.4612 F-score cur/max: 0.4242/0.4722
2021/09/25 20:59:51] Epoch: 287/300 Loss: 0.1690/0.2677/0.4367 F-score cur/max: 0.4399/0.4722
2021/09/25 20:59:52] Epoch: 288/300 Loss: 0.1680/0.2633/0.4313 F-score cur/max: 0.4267/0.4722
2021/09/25 20:59:52] Epoch: 289/300 Loss: 0.1717/0.2937/0.4654 F-score cur/max: 0.4138/0.4722
2021/09/25 20:59:53] Epoch: 290/300 Loss: 0.1757/0.2930/0.4688 F-score cur/max: 0.4133/0.4722
2021/09/25 20:59:54] Epoch: 291/300 Loss: 0.1601/0.2885/0.4486 F-score cur/max: 0.3978/0.4722
2021/09/25 20:59:55] Epoch: 292/300 Loss: 0.1754/0.2902/0.4656 F-score cur/max: 0.4264/0.4722
2021/09/25 20:59:56] Epoch: 293/300 Loss: 0.1806/0.2610/0.4417 F-score cur/max: 0.4198/0.4722
2021/09/25 20:59:57] Epoch: 294/300 Loss: 0.1637/0.2765/0.4402 F-score cur/max: 0.4020/0.4722
2021/09/25 20:59:58] Epoch: 295/300 Loss: 0.1562/0.2646/0.4208 F-score cur/max: 0.4392/0.4722
2021/09/25 20:59:58] Epoch: 296/300 Loss: 0.1679/0.3195/0.4874 F-score cur/max: 0.3818/0.4722
2021/09/25 20:59:59] Epoch: 297/300 Loss: 0.1641/0.2772/0.4414 F-score cur/max: 0.4443/0.4722
2021/09/25 20:59:59] Epoch: 298/300 Loss: 0.1668/0.2926/0.4594 F-score cur/max: 0.4032/0.4722
2021/09/25 21:00:01] Epoch: 299/300 Loss: 0.1558/0.2874/0.4432 F-score cur/max: 0.4045/0.4722
2021/09/25 21:00:01] Start training on summe: split 1
2021/09/25 21:00:02] Epoch: 0/300 Loss: 0.9570/1.1018/2.0588 F-score cur/max: 0.4446/0.4446
2021/09/25 21:00:03] Epoch: 1/300 Loss: 0.9430/0.9692/1.9122 F-score cur/max: 0.4458/0.4458
2021/09/25 21:00:04] Epoch: 2/300 Loss: 0.8813/0.8663/1.7475 F-score cur/max: 0.4368/0.4458
2021/09/25 21:00:04] Epoch: 3/300 Loss: 0.7956/0.8545/1.6501 F-score cur/max: 0.4257/0.4458
2021/09/25 21:00:05] Epoch: 4/300 Loss: 0.7778/0.8236/1.6014 F-score cur/max: 0.4153/0.4458
2021/09/25 21:00:06] Epoch: 5/300 Loss: 0.7550/0.8230/1.5780 F-score cur/max: 0.4402/0.4458
2021/09/25 21:00:07] Epoch: 6/300 Loss: 0.7383/0.7884/1.5267 F-score cur/max: 0.4359/0.4458
2021/09/25 21:00:08] Epoch: 7/300 Loss: 0.6949/0.8191/1.5140 F-score cur/max: 0.4802/0.4802
2021/09/25 21:00:09] Epoch: 8/300 Loss: 0.6799/0.7514/1.4313 F-score cur/max: 0.4714/0.4802
2021/09/25 21:00:10] Epoch: 9/300 Loss: 0.6589/0.7657/1.4246 F-score cur/max: 0.4529/0.4802
2021/09/25 21:00:11] Epoch: 10/300 Loss: 0.6636/0.8155/1.4791 F-score cur/max: 0.4501/0.4802
2021/09/25 21:00:11] Epoch: 11/300 Loss: 0.6349/0.7797/1.4147 F-score cur/max: 0.4281/0.4802
2021/09/25 21:00:12] Epoch: 12/300 Loss: 0.6767/0.7724/1.4491 F-score cur/max: 0.4224/0.4802
```

(Figure 4.12) - Training the model on TVSum and SumMe datasets



## Chapter 5: Testing

We originally split the TVSum and SumMe datasets for testing and training purposes, so now we evaluated the model by testing it against the testing dataset splits of the original datasets.

We also ran the model on the CCTV video footage and tested it on that as well. The results are given in the next section.

### 5.1 Testing details

We changed hyperparameters and fine tuned the model in order to increase its accuracy. Finally we achieved testing accuracy scores of 60.3 % on the TVSum Dataset and 48.9% on the SumMe Dataset

```
../models/custom/checkpoint/custom.yml.0.pt --source ../custom_data/videos/St_Maarten_Land  
ing.mp4 --save-path ./output.mp4  
Loading DSNet model ...  
Preprocessing source video ...  
Predicting summary ...  
Writing summary video ...  
nishit@nishit-ubuntu:~/Downloads/DSNet/src$
```

```
../custom_data/custom.yml'], 'max_epoch': 300, 'model_dir': '../models/custom', 'log_file': '  
log.txt', 'lr': 5e-05, 'weight_decay': 1e-05, 'lambda_reg': 1.0, 'nms_thresh': 0.5, 'ckpt_pat  
h': None, 'sample_rate': 15, 'source': None, 'save_path': None, 'base_model': 'attention', 'n  
um_head': 8, 'num_feature': 1024, 'num_hidden': 128, 'neg_sample_ratio': 2.0, 'incomplete_sam  
ple_ratio': 1.0, 'pos_iou_thresh': 0.6, 'neg_iou_thresh': 0.0, 'incomplete_iou_thresh': 0.3,  
'anchor_scales': [4, 8, 16, 32], 'lambda_ctr': 1.0, 'cls_loss': 'focal', 'reg_loss': 'soft-  
io
```

```
[2021/09/25 21:16:49] Epoch: 274/300 Loss: 0.1779/0.2996/0.4774 F-score cur/max: 0.3512/0.4167  
[2021/09/25 21:16:49] Epoch: 275/300 Loss: 0.1653/0.3355/0.5007 F-score cur/max: 0.3529/0.4167  
[2021/09/25 21:16:50] Epoch: 276/300 Loss: 0.1503/0.3011/0.4514 F-score cur/max: 0.3364/0.4167  
[2021/09/25 21:16:51] Epoch: 277/300 Loss: 0.1457/0.3248/0.4706 F-score cur/max: 0.3364/0.4167  
[2021/09/25 21:16:52] Epoch: 278/300 Loss: 0.1635/0.3065/0.4700 F-score cur/max: 0.3665/0.4167  
[2021/09/25 21:16:53] Epoch: 279/300 Loss: 0.1520/0.3267/0.4786 F-score cur/max: 0.3604/0.4167  
[2021/09/25 21:16:54] Epoch: 280/300 Loss: 0.1618/0.3262/0.4881 F-score cur/max: 0.3495/0.4167  
[2021/09/25 21:16:55] Epoch: 281/300 Loss: 0.1560/0.2958/0.4519 F-score cur/max: 0.3826/0.4167  
[2021/09/25 21:16:55] Epoch: 282/300 Loss: 0.1440/0.3111/0.4550 F-score cur/max: 0.3859/0.4167  
[2021/09/25 21:16:56] Epoch: 283/300 Loss: 0.1584/0.3055/0.4639 F-score cur/max: 0.3850/0.4167  
[2021/09/25 21:16:57] Epoch: 284/300 Loss: 0.1332/0.3256/0.4588 F-score cur/max: 0.3608/0.4167  
[2021/09/25 21:16:58] Epoch: 285/300 Loss: 0.1483/0.2727/0.4210 F-score cur/max: 0.3587/0.4167  
[2021/09/25 21:16:59] Epoch: 286/300 Loss: 0.1482/0.3167/0.4649 F-score cur/max: 0.3326/0.4167  
[2021/09/25 21:17:00] Epoch: 287/300 Loss: 0.1486/0.2995/0.4481 F-score cur/max: 0.3435/0.4167  
[2021/09/25 21:17:00] Epoch: 288/300 Loss: 0.1668/0.3013/0.4682 F-score cur/max: 0.3248/0.4167  
[2021/09/25 21:17:01] Epoch: 289/300 Loss: 0.1498/0.2853/0.4351 F-score cur/max: 0.3289/0.4167  
[2021/09/25 21:17:02] Epoch: 290/300 Loss: 0.1517/0.2907/0.4424 F-score cur/max: 0.3589/0.4167  
[2021/09/25 21:17:03] Epoch: 291/300 Loss: 0.1404/0.3141/0.4545 F-score cur/max: 0.3497/0.4167  
[2021/09/25 21:17:04] Epoch: 292/300 Loss: 0.1450/0.2959/0.4409 F-score cur/max: 0.3415/0.4167  
[2021/09/25 21:17:05] Epoch: 293/300 Loss: 0.1614/0.2837/0.4451 F-score cur/max: 0.3285/0.4167  
[2021/09/25 21:17:06] Epoch: 294/300 Loss: 0.1435/0.2835/0.4271 F-score cur/max: 0.3657/0.4167  
[2021/09/25 21:17:07] Epoch: 295/300 Loss: 0.1562/0.2933/0.4494 F-score cur/max: 0.3614/0.4167  
[2021/09/25 21:17:07] Epoch: 296/300 Loss: 0.1419/0.2966/0.4385 F-score cur/max: 0.3328/0.4167  
[2021/09/25 21:17:08] Epoch: 297/300 Loss: 0.1567/0.3053/0.4620 F-score cur/max: 0.3422/0.4167  
[2021/09/25 21:17:09] Epoch: 298/300 Loss: 0.1513/0.2942/0.4455 F-score cur/max: 0.3343/0.4167  
[2021/09/25 21:17:10] Epoch: 299/300 Loss: 0.1346/0.2852/0.4198 F-score cur/max: 0.3438/0.4167  
[2021/09/25 21:17:10] Training done on summe. F-score: 0.4890  
nishit@nishit-ubuntu:~/Downloads/DSNet/src$
```

```

[2021/09/25 20:55:05] Epoch: 284/300 Loss: 0.1102/0.0260/0.1302 F-score cur/max: 0.4425/0.6010
[2021/09/25 20:55:06] Epoch: 285/300 Loss: 0.1120/0.0267/0.1386 F-score cur/max: 0.4272/0.6010
[2021/09/25 20:55:08] Epoch: 286/300 Loss: 0.1128/0.0247/0.1375 F-score cur/max: 0.4254/0.6010
[2021/09/25 20:55:10] Epoch: 287/300 Loss: 0.1135/0.0254/0.1390 F-score cur/max: 0.4305/0.6010
[2021/09/25 20:55:13] Epoch: 288/300 Loss: 0.1116/0.0244/0.1360 F-score cur/max: 0.4333/0.6010
[2021/09/25 20:55:15] Epoch: 289/300 Loss: 0.1197/0.0235/0.1432 F-score cur/max: 0.3965/0.6010
[2021/09/25 20:55:18] Epoch: 290/300 Loss: 0.1114/0.0246/0.1360 F-score cur/max: 0.4290/0.6010
[2021/09/25 20:55:20] Epoch: 291/300 Loss: 0.1144/0.0239/0.1383 F-score cur/max: 0.4026/0.6010
[2021/09/25 20:55:23] Epoch: 292/300 Loss: 0.1259/0.0246/0.1505 F-score cur/max: 0.4176/0.6010
[2021/09/25 20:55:25] Epoch: 293/300 Loss: 0.1437/0.0241/0.1678 F-score cur/max: 0.4522/0.6010
[2021/09/25 20:55:28] Epoch: 294/300 Loss: 0.1249/0.0239/0.1488 F-score cur/max: 0.4196/0.6010
[2021/09/25 20:55:30] Epoch: 295/300 Loss: 0.1219/0.0242/0.1461 F-score cur/max: 0.4162/0.6010
[2021/09/25 20:55:33] Epoch: 296/300 Loss: 0.1039/0.0229/0.1268 F-score cur/max: 0.4270/0.6010
[2021/09/25 20:55:35] Epoch: 297/300 Loss: 0.1092/0.0235/0.1327 F-score cur/max: 0.4227/0.6010
[2021/09/25 20:55:37] Epoch: 298/300 Loss: 0.1048/0.0236/0.1284 F-score cur/max: 0.4110/0.6010
[2021/09/25 20:55:40] Epoch: 299/300 Loss: 0.1048/0.0228/0.1276 F-score cur/max: 0.3967/0.6010
[2021/09/25 20:55:40] Training done on tvsum. F-score: 0.6030
[2021/09/25 20:55:40] Start training on summe: split 0
[2021/09/25 20:55:41] Epoch: 0/300 Loss: 1.0968/1.1236/2.2204 F-score cur/max: 0.4688/0.4688
[2021/09/25 20:55:42] Epoch: 1/300 Loss: 0.9517/0.9833/1.9351 F-score cur/max: 0.4529/0.4688
[2021/09/25 20:55:42] Epoch: 2/300 Loss: 0.9074/0.8053/1.7126 F-score cur/max: 0.4439/0.4688
[2021/09/25 20:55:43] Epoch: 3/300 Loss: 0.8436/0.8475/1.6911 F-score cur/max: 0.4722/0.4722
[2021/09/25 20:55:44] Epoch: 4/300 Loss: 0.8505/0.7720/1.6225 F-score cur/max: 0.4607/0.4722
[2021/09/25 20:55:45] Epoch: 5/300 Loss: 0.7988/0.8043/1.6032 F-score cur/max: 0.4378/0.4722
[2021/09/25 20:55:46] Epoch: 6/300 Loss: 0.7301/0.7322/1.4623 F-score cur/max: 0.4378/0.4722
[2021/09/25 20:55:47] Epoch: 7/300 Loss: 0.7228/0.7744/1.4972 F-score cur/max: 0.4436/0.4722
[2021/09/25 20:55:48] Epoch: 8/300 Loss: 0.6863/0.8255/1.5118 F-score cur/max: 0.4378/0.4722
[2021/09/25 20:55:48] Epoch: 9/300 Loss: 0.6972/0.7775/1.4748 F-score cur/max: 0.4245/0.4722

```

(Figure 5.1) - Testing our model on TVSum and SumMe Datasets

## 5.2 Issues and limitations of the solution :

Issues faced by us during the implementation were-

1. Some of the videos in the datasets were too hazy and of bad quality, which reduced the accuracy of the model. We had to manually remove these videos from training data.
2. Some of the human created ground truth summaries of the videos are not temporally coherent and thus are not good at all. Thus when comparing Temporal IOU, the results were not that good as the ground truth itself was not a very good summary.
3. As these are large videos and datasets we are dealing with, it requires a lot of resources and GPU Memory. Thus many times, while training the model, we ran out of GPU memory and had to start training the model from the start.



## Chapter 6 : Observation and Results

### 6.1 Findings

We changed hyperparameters and fine tuned the model in order to increase its accuracy. Finally we achieved testing accuracy scores of 60.3 % on the TVSum Dataset and 48.9% on the SumMe Dataset. Here is our F-Score as compared to the work of some previous authors on the topic of Video Summarisation on TVSum and SumMe Datasets.

| Model                    | TVSum Accuracy | SumMe Accuracy |
|--------------------------|----------------|----------------|
| FCSN(Rochan et al.)      | 52.7           | 41.5           |
| VASNet (J. Fajtl et al.) | 59.8           | 47.7           |
| DR-DSN (Zhou et al.)     | 57.6           | 41.4           |
| Ours                     | 60.3           | 48.9           |

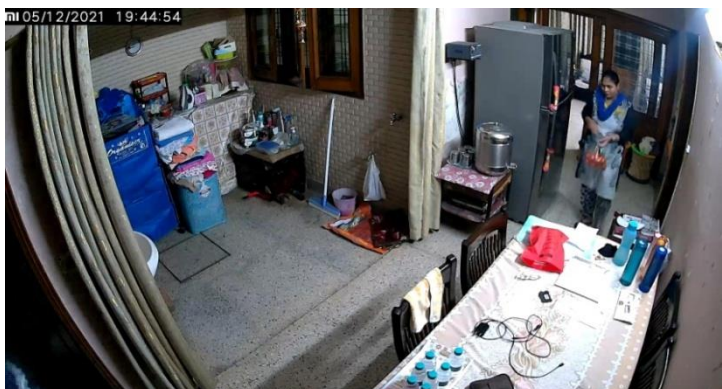
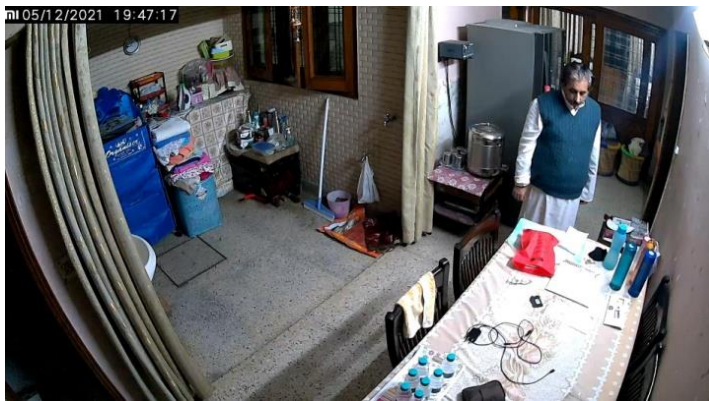
(Table 6.1): Our Final F-Score as compared to previous work done in this field

```
21:17:07] Epoch: 296/300 Loss: 0.1419/0.2966/0.4385
21:17:08] Epoch: 297/300 Loss: 0.1567/0.3053/0.4620
21:17:09] Epoch: 298/300 Loss: 0.1513/0.2942/0.4455
21:17:10] Epoch: 299/300 Loss: 0.1346/0.2852/0.4198
21:17:10] Training done on summe. F-score: 0.4890
```

```
20:55:37] Epoch: 298/300 Loss: 0.1048/0.0236/0.1284
20:55:40] Epoch: 299/300 Loss: 0.1048/0.0228/0.1276
20:55:40] Training done on tvsum. F-score: 0.6030
```

(Figure 6.1) - Final F-Scores of our model on TVSum and SumMe Datasets





(Figure 6.2) - Screenshots from summary video created by our model on CCTV surveillance footage

## 6.2 Gantt Chart

|  |   |  |  |
|--|---|--|--|
| Make variations to the model to find the best deep learning layer suitable | Try to increase the temporal coherence between the scenes | Increase the Temporal IOU by finding the most suitable value of Temporal IOU Threshold | Fine tune the model to increase the Test Accuracy of the model |
|--|---|--|--|

January

February

March

April

May

(Table 6.2) - Gantt Chart for future plan

## 6.3 Future Plan and Conclusion

Thus, this report represents our framework for video summarization. Unlike existing supervised methods which only learn the importance score of each frame, our approach formulates video summarization as a temporal interest detection problem and simultaneously helps to learn temporal coherence between scenes in a video while handling incorrect and incomplete segments. In the future, we will attempt to incorporate more interest based coherence into our unified framework.



(Figure 6.3) - Few more screenshots from summary video created by our model on CCTV surveillance footage

## References

- [1] S.-H. T. H.-T. C. Tsu-Jui Fu, "Attentive and adversarial learning for video summarization," *Attentive and adversarial learning for video summarization*, 2019.
- [2] J. A. G. S. H. R. E. Hussain Kanafani, "Unsupervised Video Summarization via Multi-source Features.," *Unsupervised Video Summarization via Multi-source Features.*, 2021.
- [3] E. A. A. I. M. V. M. a. I. P. E. Apostolidis, "AC-SUM-GAN: Connecting Actor-Critic and Generative Adversarial Networks for Unsupervised Video Summarization," *AC-SUM-GAN: Connecting Actor-Critic and Generative Adversarial Networks for Unsupervised Video Summarization*, vol. 31, no. 8, pp. 3278-3292, 2021.
- [4] J. L. L. Z. Wencheng Zhu, "DSNet: A Flexible Detect-to-Summarize Network for Video Summarization," *DSNet: A Flexible Detect-to-Summarize Network for Video Summarization*, pp. 948 - 962, 2020.
- [5] R. P. Q. Z. ., K. R.-C. Shuyue Lan, "FFNet: Video Fast-Forwarding via Reinforcement Learning," *FFNet: Video Fast-Forwarding via Reinforcement Learning*, 2018.
- [6] Y. Q. T. X. Kaiyang Zhou, "Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity," *Deep Reinforcement Learning for Unsupervised Video Summarization with Diversity*, 2018.
- [7] F. K. M. L. Pinelopi Papalampidi, "Movie Summarization via Sparse Graph Construction," *Pinelopi Papalampidi Frank Keller Mirella LapataMovie Summarization via Sparse Graph Construction*, 2020.

- [8] S. S. I. S. J. S. Fred Hohman, "A Viz of Ice and Fire: Exploring Entertainment Video Using Color and Dialogue," *A Viz of Ice and Fire: Exploring Entertainment Video Using Color and Dialogue*, 2017.
- [9] M. G. A. V. Arun Balajee Vasudevan, "Query-adaptive Video Summarization via Quality-aware Relevance Estimation.," *Query-adaptive Video Summarization via Quality-aware Relevance Estimation.*, 2017.
- [10] L. Y. a. Y. W. Mrigank Rochan, "Video Summarization Using Fully Convolutional," *Video Summarization Using Fully Convolutional*, 2018.
- [11] H. S. S. V. A. D. M. a. P. R. Jiri Fajtl, "Summarizing Videos with Attention," *Summarizing Videos with Attention*, 2019.
- [12] M. J. Shruti Jadon, "Unsupervised video summarization framework using keyframe extraction and video skimming.," *Unsupervised video summarization framework using keyframe extraction and video skimming.*, 2020.
- [13] D. R. a. N. S. Yair Shemer, "ILS-SUMM: Iterated local search for unsupervised video summarization," *ILS-SUMM: Iterated local search for unsupervised video summarization*, 2019.
- [14] M. W. Jia-Hong Huang, "Query controllable video summarization," *Query controllable video summarization*, 2019.
- [15] L. W. a. A. B. Mohamed Elfeki, "Multi stream dynamic video summarization," *Multi stream dynamic video summarization*, 2019.