Video Summarization

By
Nishit Anand (9918103133)
Shreya Agarwal (9918103146)
Aditi Dixit (9918103155)

Abstract

With the ever-increasing popularity and decreasing cost of video capture devices, the amount of video data has increased drastically in the past few years. Video has become one of the most important form 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 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.

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. 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.

Significance

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. Dependent on application and the target audience, the evaluation of summary often involves usability studies to measure the content informativeness and quality of the summary.

Background study

We studied the literature pertaining to the various video summarization techniques by going through the existing research papers. We studied various techniques by other authors for video summarisation.

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 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 Neural Networks and Convolutional Neural Networks. This requires a good amount of computational power and GPU power to do the required ML tasks quickly and efficiently.

Requirement Analysis

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.
- For validation purposes, we also need testing data, so that we can check the performance of our model.

Data Sets used

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

TVSum - Title-based Video Summarization (TVSum) dataset serves as a benchmark to validate video summarization techniques. It contains 50 videos of various genres (e.g., news, how-to, documentary, vlog, egocentric) and 1,000 annotations of shot-level importance scores obtained via crowdsourcing (20 per video).

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.

Detailed Description of solution approach

- **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.

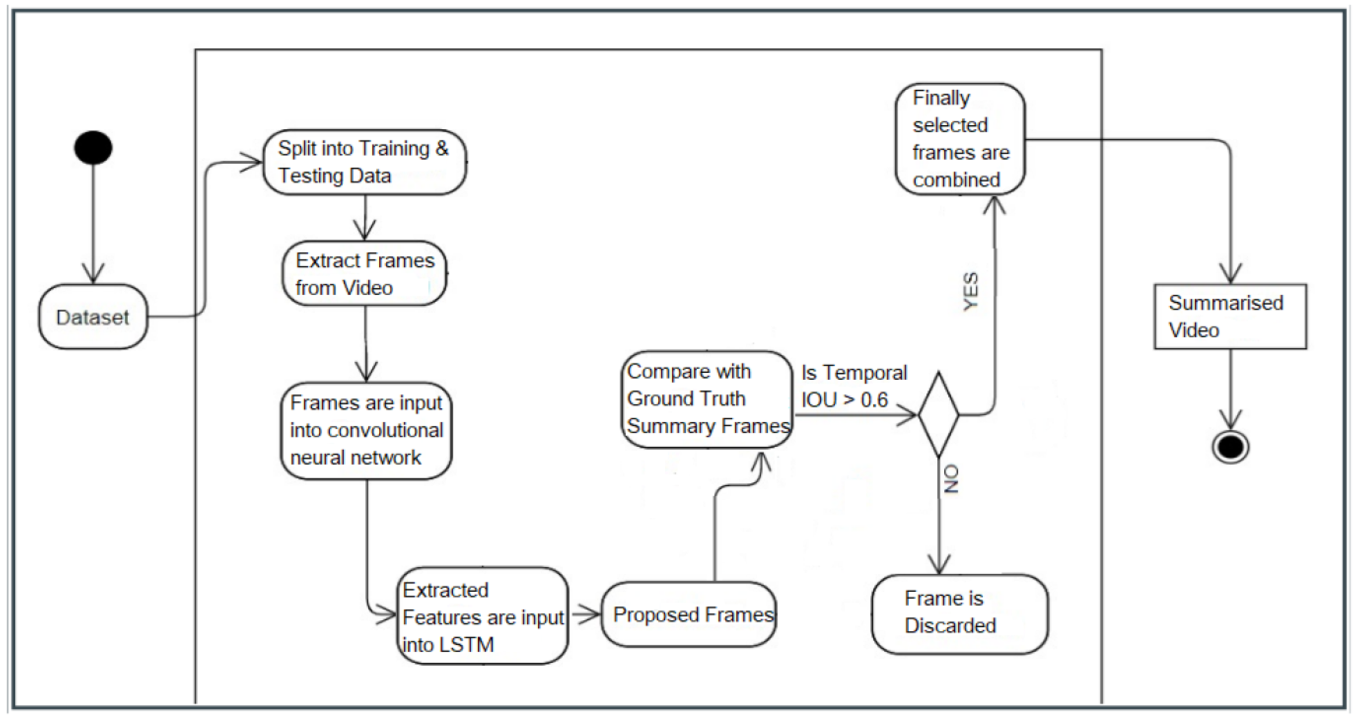
- **Feature Extraction from the extracted frames:** Our model needs to learn the features from these extracted frames. For this, we use convolutional neural networks to extract features from the video frames. Now, our model starts learning from the frames extracted from the video in the above step. Moreover, 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.

- **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.

- **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.

- **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.

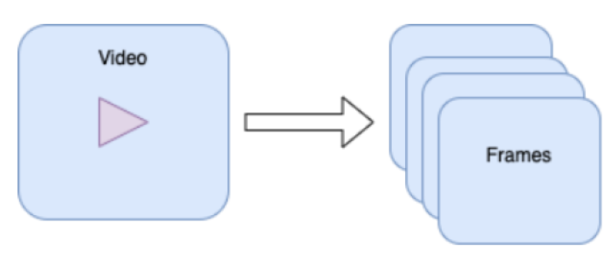
Implementation Details : FlowChart


Dataset
Split into Training & Testing Data
Extract Frames from Video
Frames are input into convolutional neural network
Extracted Features are input into LSTM
Proposed Frames
Compare with Ground Truth Summary Frames
Is Temporal IOU > 0.6
YES
NO
Frame is Discarded
Finally selected frames are combined
Summarised Video

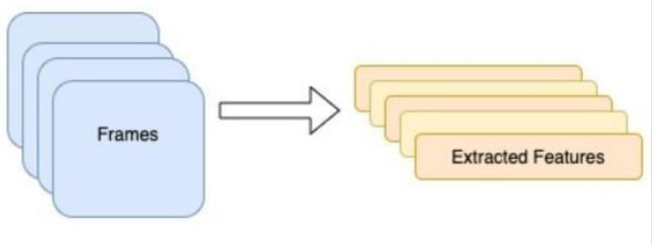
Implementation Details: Steps

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 corresponding 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.

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

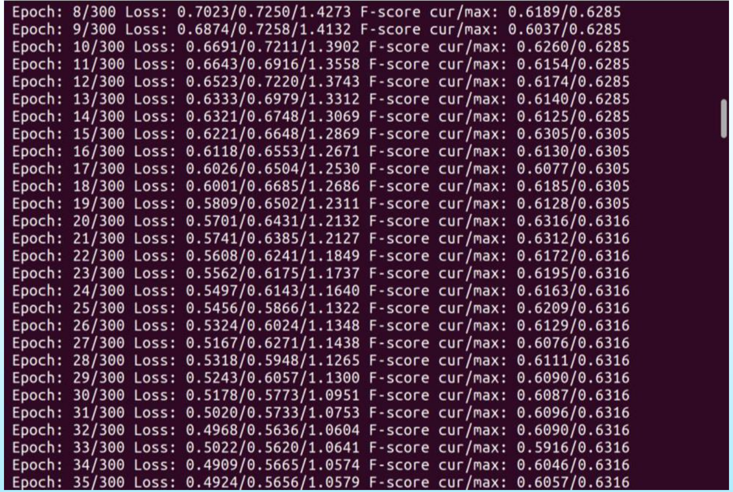


3. Detect and Extract Features:
We wrote a custom Machine learning model made up of Convolutional layers and dropout layers in between them. The Convolutional Neural Network extracts spatial features from the images and trains on them, so that it can better predict the right frame to be selected for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

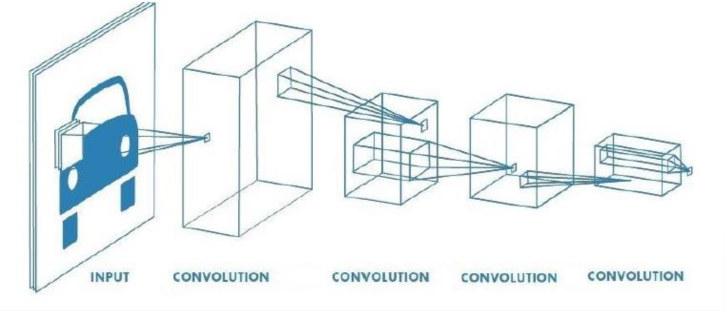


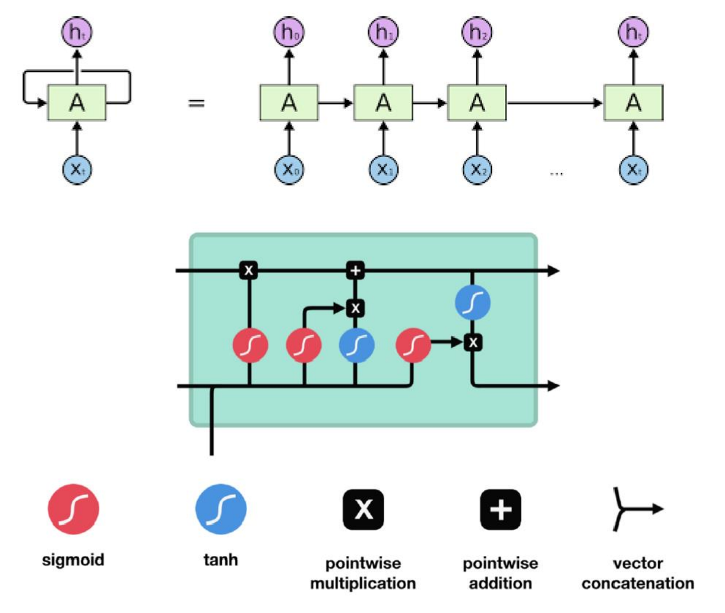
4. Reducing Overfitting:

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



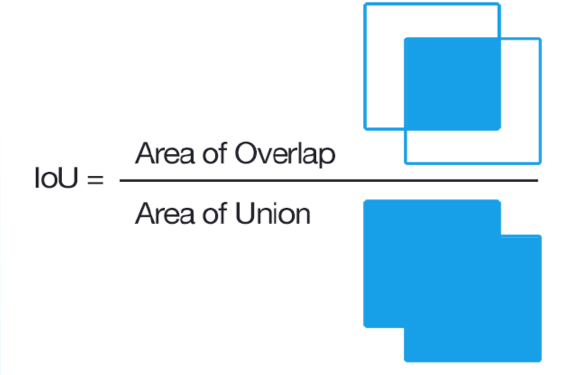
```
Epoch: 8/300 Loss: 0.7023/0.7250/1.4273 F-score cur/max: 0.6189/0.6285
Epoch: 9/300 Loss: 0.6874/0.7258/1.4132 F-score cur/max: 0.6037/0.6285
Epoch: 10/300 Loss: 0.6691/0.7211/1.3902 F-score cur/max: 0.6260/0.6285
Epoch: 11/300 Loss: 0.6643/0.6916/1.3558 F-score cur/max: 0.6154/0.6285
Epoch: 12/300 Loss: 0.6523/0.7220/1.3743 F-score cur/max: 0.6174/0.6285
Epoch: 13/300 Loss: 0.6333/0.6979/1.3312 F-score cur/max: 0.6140/0.6285
Epoch: 14/300 Loss: 0.6321/0.6748/1.3069 F-score cur/max: 0.6125/0.6285
Epoch: 15/300 Loss: 0.6221/0.6648/1.2869 F-score cur/max: 0.6305/0.6305
Epoch: 16/300 Loss: 0.6118/0.6553/1.2671 F-score cur/max: 0.6130/0.6305
Epoch: 17/300 Loss: 0.6026/0.6504/1.2530 F-score cur/max: 0.6077/0.6305
Epoch: 18/300 Loss: 0.6001/0.6685/1.2686 F-score cur/max: 0.6185/0.6305
Epoch: 19/300 Loss: 0.5809/0.6502/1.2311 F-score cur/max: 0.6128/0.6305
Epoch: 20/300 Loss: 0.5701/0.6431/1.2132 F-score cur/max: 0.6316/0.6316
Epoch: 21/300 Loss: 0.5741/0.6385/1.2127 F-score cur/max: 0.6312/0.6316
Epoch: 22/300 Loss: 0.5608/0.6241/1.1849 F-score cur/max: 0.6172/0.6316
Epoch: 23/300 Loss: 0.5562/0.6175/1.1737 F-score cur/max: 0.6195/0.6316
Epoch: 24/300 Loss: 0.5497/0.6143/1.1640 F-score cur/max: 0.6163/0.6316
Epoch: 25/300 Loss: 0.5456/0.5866/1.1322 F-score cur/max: 0.6209/0.6316
Epoch: 26/300 Loss: 0.5324/0.6024/1.1348 F-score cur/max: 0.6129/0.6316
Epoch: 27/300 Loss: 0.5167/0.6271/1.1438 F-score cur/max: 0.6076/0.6316
Epoch: 28/300 Loss: 0.5318/0.5948/1.1265 F-score cur/max: 0.6111/0.6316
Epoch: 29/300 Loss: 0.5243/0.6057/1.1300 F-score cur/max: 0.6090/0.6316
Epoch: 30/300 Loss: 0.5178/0.5773/1.0951 F-score cur/max: 0.6087/0.6316
Epoch: 31/300 Loss: 0.5020/0.5733/1.0753 F-score cur/max: 0.6096/0.6316
Epoch: 32/300 Loss: 0.4968/0.5636/1.0604 F-score cur/max: 0.6090/0.6316
Epoch: 33/300 Loss: 0.5022/0.5620/1.0641 F-score cur/max: 0.5916/0.6316
Epoch: 34/300 Loss: 0.4909/0.5665/1.0574 F-score cur/max: 0.6046/0.6316
Epoch: 35/300 Loss: 0.4924/0.5656/1.0579 F-score cur/max: 0.6057/0.6316
```

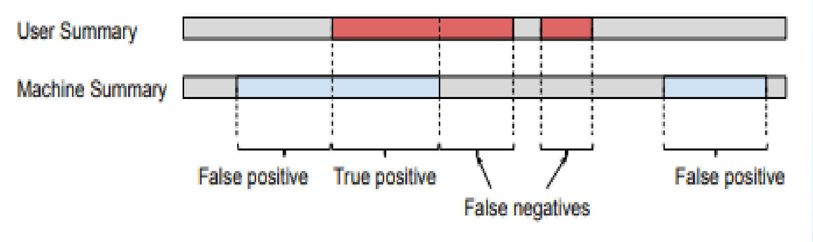


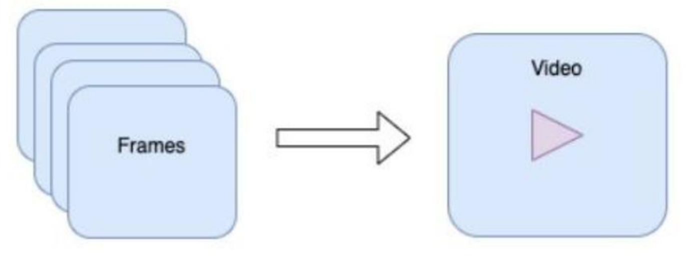


5. Feeding these frames into LSTM:

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 summary video. These we call as temporal interest proposal frames







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

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

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

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

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 OpenCV python library for this.

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.

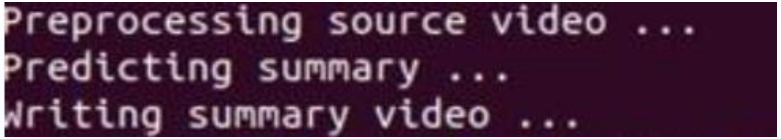
After this the model became robust and was able to find which parts of the CCTV video footage have humans and the model was able to separate them out pretty well from the rest of the clip and able to create a good summary video containing all the parts having human activity.

11**. Testing the model:**

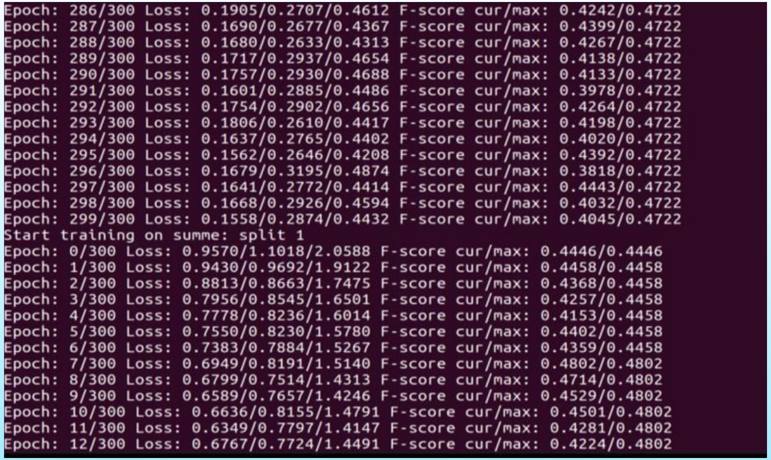
We originally split the TVSum and SumMe datasets for testing and training purposes, so now we evaluated the model by testing it against the testing dataset splits of the original datasets.

12. Fine Tuning the Model:

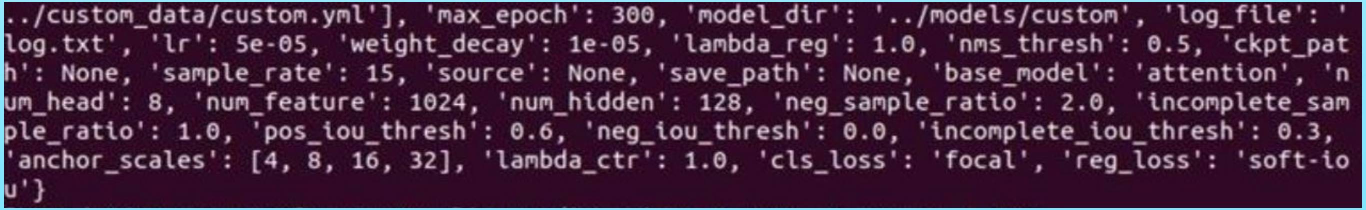
We changed hyperparameters and fine tuned the model in order to increase its accuracy. Finally we achieved testing accuracy scores of 60.3 % on the TVSum Dataset and 48.9% on the SumMe Dataset.



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



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

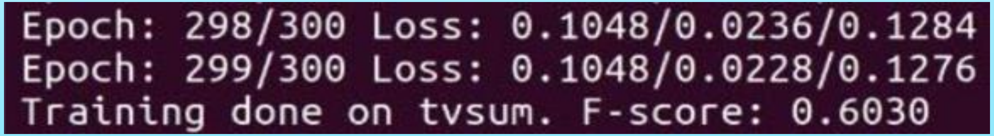


```
../custom_data/custom.yml'], 'max_epoch': 300, 'model_dir': '../models/custom', 'log_file': '
log.txt', 'lr': 5e-05, 'weight_decay': 1e-05, 'lambda_reg': 1.0, 'nms_thresh': 0.5, 'ckpt_pat
h': None, 'sample_rate': 15, 'source': None, 'save_path': None, 'base_model': 'attention', 'n
um_head': 8, 'num_feature': 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
u'}
```

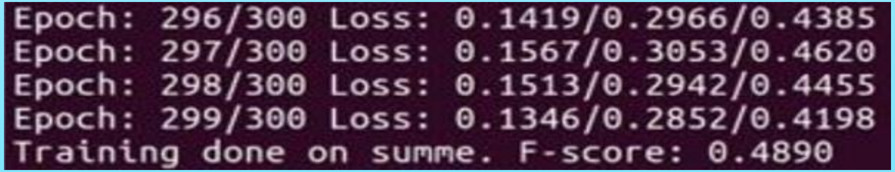
Results

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
VASNet (J. Fajtl et al.)	61.42	49.71
Rochan et al.	52.7	41.5
DR-DSN (Zhou et al.)	57.6	41.4
Ours	60.3	48.9

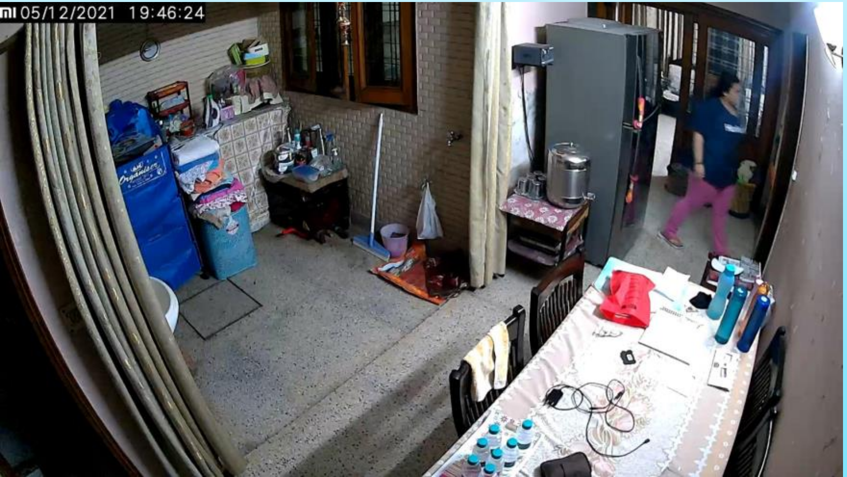


```
Epoch: 298/300 Loss: 0.1048/0.0236/0.1284
Epoch: 299/300 Loss: 0.1048/0.0228/0.1276
Training done on tvsum. F-score: 0.6030
```



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

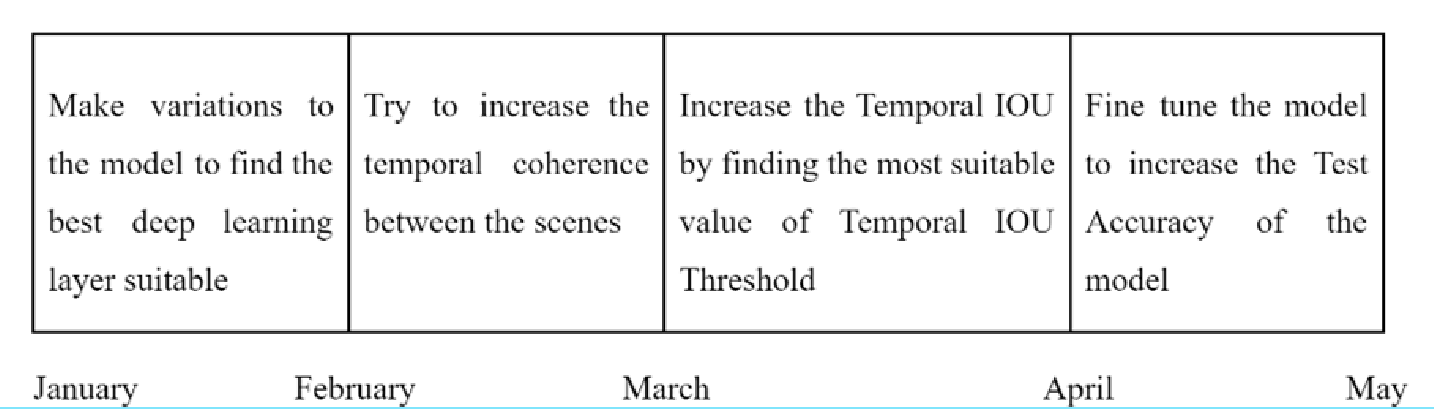
Results - Some examples of our model detecting humans in CCTV video footage



Conclusion and Future Scope

Thus, this is our idea to create a 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.

Future Plan and 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