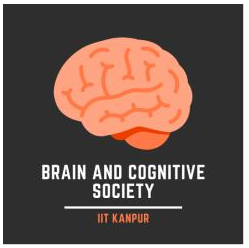


Semester Projects
Science and Technology Council
IIT Kanpur

Convolutional Network for Online Video Understanding

Documentation

Date of Submission:

June 25, 2021

Contents

1)Acknowledgements

We are immensely grateful to Brain And Cognitive Society, IIT Kanpur, and Science & Technology Council, IIT Kanpur for providing us with the resources and motivation for this Summer Project. We thank our mentors: Tanvi Nerkar, Shivangi Singh; the Coordinators of Brain And Cognitive Society: Shivanshu Tayagi, Mohit Kulkarni for their guidance, constant support, and invaluable inputs, without which, this project would not have been possible.

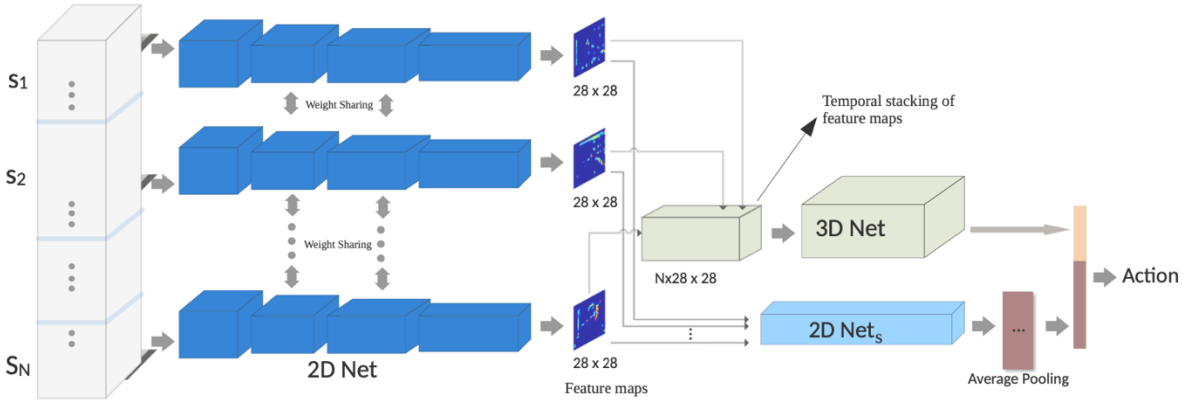
2)Introduction

- Aim

This project aims to build CNN Model for Video Understanding using kinetics600 & UCF101 data sets. We've also implemented Video Captioning within a video, where each video can contain up to several hundred frames.

- Abstract

The state of the art in video understanding suffers from two problems: (1) The major part of the reasoning is performed locally in the video, therefore, it misses important relationships within actions that span several seconds. (2) While there are local methods with fast per-frame processing, the processing of the whole video is not efficient and hampers fast video retrieval or online classification of long-term activities. In this paper, we introduce a network architecture that takes long-term content into account and enables fast per-video processing at the same time. The architecture is based on merging long-term content already in the network rather than in a post-hoc fusion. Together with a sampling strategy, which exploits that neighboring frames are largely redundant, this yields high-quality action classification and video captioning at up to 230 videos per second, where each video can consist of a few hundred frames. The approach achieves competitive performance across all datasets while being 10x to 80x faster than state-of-the-art methods.



Architecture Overview of ECO

3)Brief Approach

Each video is split into N subsections of equal size. From each subsection, a single frame is randomly sampled. The samples are processed by a regular 2D convolutional network to yield a representation for each sampled frame. In this design, we use a 2D network in parallel with the 3D network. 2D-Net directly provides the visual semantics of single frames and 3D net processes the temporally stacked representations of frames using a 3D convolutional network. We apply average pooling for the 2D network to generate video-level features and concatenate them with the features from 3D-net. For simplicity, the figure just shows one channel of the 96 output channels of 2D-Net.

4)Related Work

- Video Classification with deep learning

Most recent works on video classification are based on deep learning. To explore the temporal context of a video, 3D convolutional networks are on obvious option use of a Resnet architecture with 3D convolutions was studied and it showed the improvements over their earlier c3d architecture. An alternative way to model the temporal relation between frames is by using recurrent networks. However, the performance of recurrent networks on action recognition currently lags behind that of recent CNN-based methods, which may indicate that they do not sufficiently model long-term dynamics. Recently, several works utilized 3D architectures for action recognition. These approaches model the short-term temporal context of the input video based on a sliding window. At inference time, these methods must compute the average score over multiple windows, which is quite time-consuming. For example, ARTNet requires on average 250 samples to classify one video.

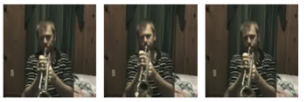
All these approaches do not sufficiently use the comprehensive information from the entire video during training and inference. Partial observation not only confuses action prediction but also requires an extra post-processing step to fuse scores. Extra feature/score aggregation reduces the speed of video processing and disables the method to work in a real-time setting.

- Long-Term Representation Learning

To cope with the problem of partial observation, some methods increased the temporal resolution of the sliding window. However, expanding the temporal length of the input has two major drawbacks. (1) It is computationally expensive, and (2) still fails to cover the visual information of the entire video, especially for longer videos.
TSN employed a sparse and global temporal sampling method to choose frames from the entire video during training. However, as in the aggregation methods above, frames are processed independently at inference time and their scores are aggregated only in the end. Consequently, the performance in their experiments stays the same when they change the number of samples, which indicates that their model does not benefit from the long-range temporal information.
Our work is different from these previous approaches in three main aspects. Therefore, it can be run in online mode and in real-time even on small computing devices.

- Video Captioning

Video captioning is a widely studied problem in computer vision. Most approaches use a CNN pre-trained on image classification or action recognition to generate features. These methods, like the video understanding methods described above, utilize a frame-based feature aggregation or a sliding window over the whole video to generate video-level features. The features are then passed to a recurrent neural network to generate the video captions via a learned language model. The extracted visual features should represent both the temporal structure of the video and the static semantics of the scene.

		
SCN: a man is playing a guitar **ECO**$_L$: a man is playing a keyboard **ECO**: a man is playing a piano **ECO**$_R$: a man is playing a piano	**SCN**: a man is singing **ECO**$_L$: a man is riding a scooter **ECO**: a man is riding a bike **ECO**$_R$: a man is riding a bicycle	**SCN**: a boy is playing the music **ECO**$_L$: a boy is playing a trumpet **ECO**: a boy is playing a trumpet **ECO**$_R$: a boy is playing a trumpet
		
SCN: a man is kicking a soccer ball **ECO**$_L$: two men are fighting **ECO**: a man is attacking a man **ECO**$_R$: two men are fighting	**SCN**: a woman is mixing some meat **ECO**$_L$: a woman is seasoning a piece of meat **ECO**: a woman is mixing flour **ECO**$_R$: a woman is coating flour	**SCN**: a boy is running **ECO**$_L$: a boy is walking **ECO**: a man is doing exercise **ECO**$_R$: a man is exercising

5)Dataset

This section gives a detailed description of the kinetics600 and UCF101 dataset used in this project.

Kinetics600

It describes an extension of the DeepMind Kinetics human action dataset from 400 classes, each with at least 400 video clips, to 600 classes, each with at least 600 video clips. To scale up the dataset, we changed the data collection process so it uses multiple queries per class, with some of them in a language other than English -- Portuguese. This paper details the changes between the two versions of the dataset and includes a comprehensive set of statistics of the new version as well as baseline results using the I3D neural network architecture. The paper is a companion to the release of the ground truth labels for the public test set.
The new form of the dataset, called Kinetics-600, follows similar standards as Kinetics-400. The clips are mined from YouTube. Each video lasts for 10s. Classes are now increased from 400 to 600 as one of the main objectives of the kinetics dataset was to replicate the ImageNet dataset with 1000 classes.
The general procedure for data collection is the same as in Kinetics 400. In other words, a rundown of class names is made, at that point, a rundown of applicant YouTube URLs is acquired for each class name, and applicant 10s clasps are inspected from the recordings. These clips are used by Amazon Mechanical Turk tools that check whether those clips contain the activity class that they should. Finally, the dataset was refined as it may contain some common video.
The current state of the art is LGD-3D Two Stream. It performed well on the dataset with an accuracy of 96%.

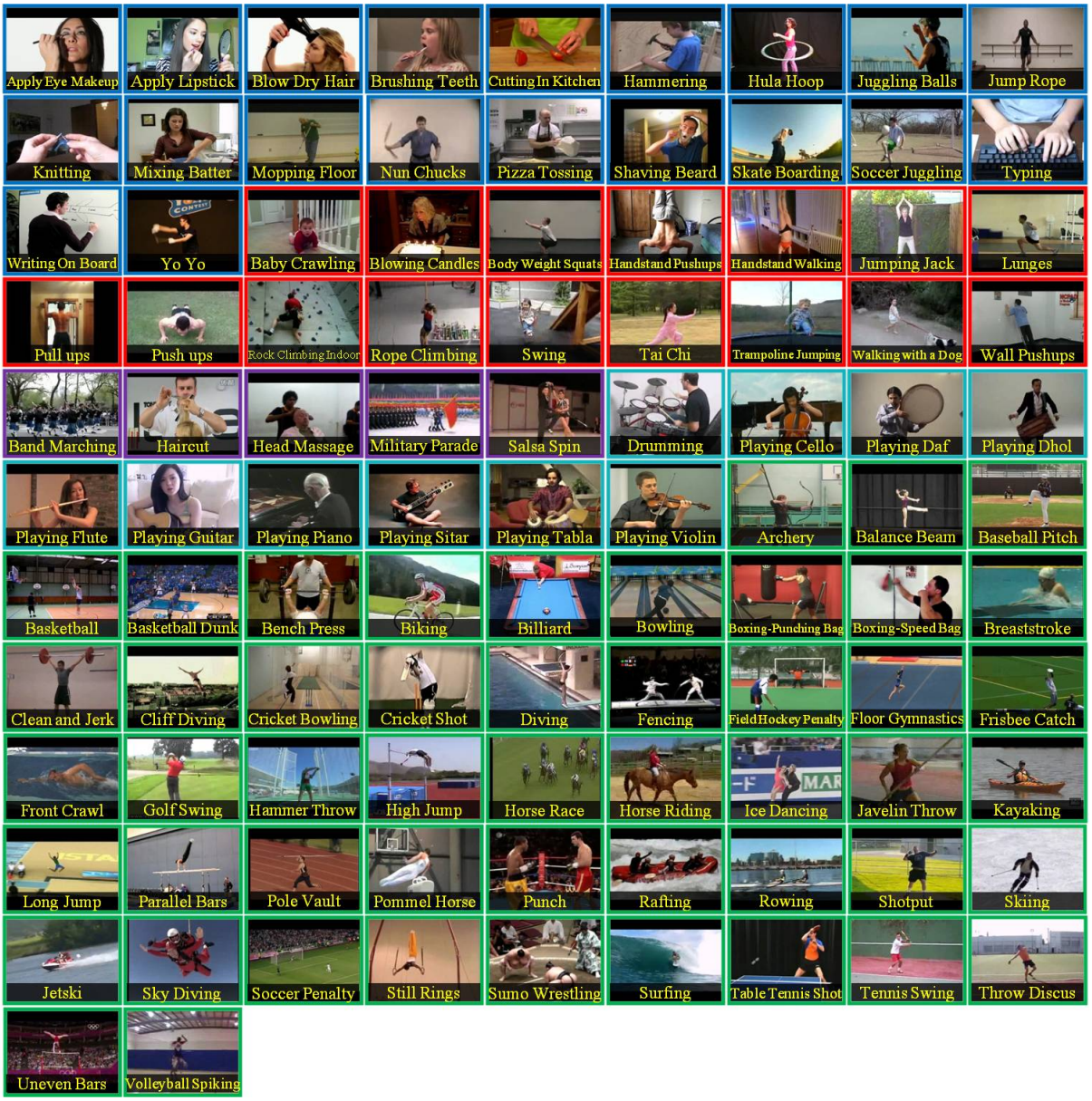
The link for the kinetics600 dataset is here

UCF101

UCF here stands for University of Central Florida. UCF101 is an action recognition data set of realistic action videos, collected from YouTube, having 101 action categories. This data set is an extension of UCF50 data set which has 50 action categories. With 13320 videos from 101 action categories, UCF101 gives the largest diversity in terms of actions and with the presence of large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, etc, it is the most challenging data set to date. As most of the available action recognition data sets are not realistic and are staged by actors, UCF101 aims to encourage further research into action recognition by learning and exploring new realistic action categories.

The videos in 101 action categories are grouped into 25 groups, where each group can consist of 4-7 videos of an action. The videos from the same group may share some common features, such as similar background, similar viewpoint, etc.

The action categories can be divided into five types: 1)Human-Object Interaction 2) Body-Motion Only 3) Human-Human Interaction 4) Playing Musical Instruments 5) Sports.

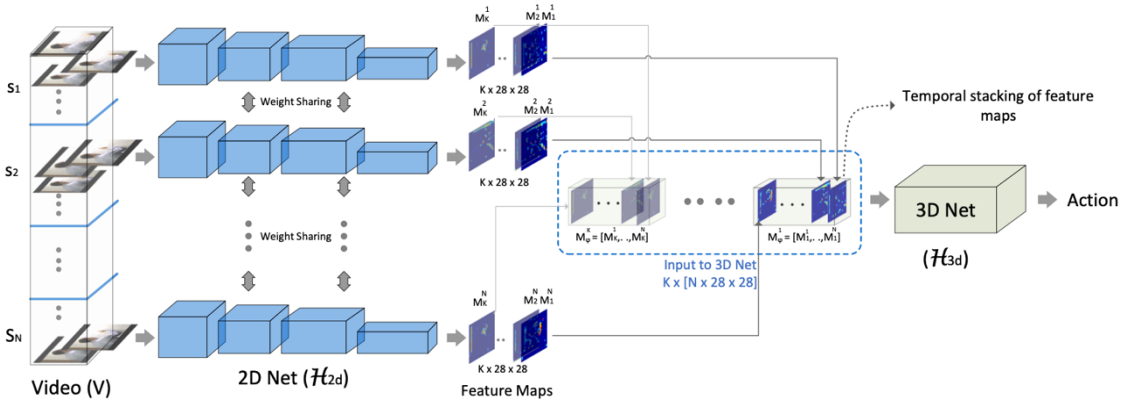


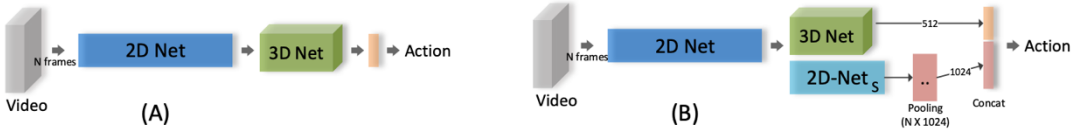
The link for the UCF101 dataset is [here]

6)Long-term Spatio-temporal Architecture

- ECO Lite and ECO Full

The 3D architecture in ECO Lite is optimized for learning relationships between the frames, but it tends to waste capacity in the case of simple short-term actions that can be recognized just from the static image content. Therefore, we suggest an extension of the architecture by using a 2D network in parallel; For the simple actions, this 2D network architecture can simplify processing and ensure that the static image features receive the necessary importance, whereas the 3D network architecture takes care of the more complex actions that depend on the relationship between frames. The 2D network receives feature maps of all samples and produces N feature representations. Afterward, we apply average pooling to get a feature vector that is representative of static scene semantics.





- Network Details
  - 2D-Net

For the 2D network (H2D) that analyses the single frames, we use the first part of the architecture. We chose this architecture due to its efficiency. The output of H2D for every single frame consists of 96 feature maps with the size of 28 × 28.

  - 3D-Net

For the 3D network H3D, we adopt several layers of 3D-Resnet18, which is an efficient architecture used in many video classification works. The output of H3D is a one-hot vector for the different class labels.

  - 2D-NetS

In the ECO full design, we use 2D-Nets in parallel with 3D-net to directly providing static visual semantics of video. For this network, we use the BN-Inception architecture from the inception-4a layer until the last pooling layer. The last pooling layer will produce a 1024 dimensional feature vector for each frame. We apply average pooling to generate video-level features and then concatenate with features obtained from 3D-net.

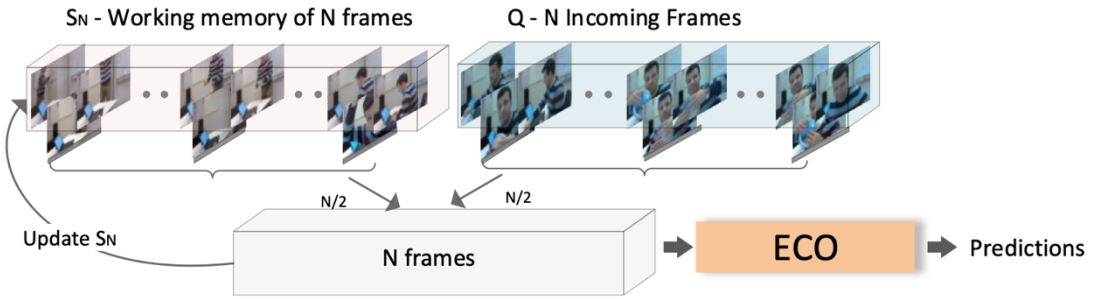
2D-Net	3D-Net	2D-Net$_S$
For the 2D network (H_{2D}) that analyzes the single frames, we use the first part of the BN-Inception architecture (until inception-3c layer) .	For the 3D network H_{3D} we adopt several layers of 3D-Resnet18, which is an efficient architecture used in many video classification works .	For this network, we use the BN-Inception architecture from inception-4a layer until last pooling layer .
The output of H_{2D} for each single frame consist of 96 feature maps with size of 28 × 28.	The out- put of H_{3D} is a one-hot vector for the different class labels.	The last pooling layer will produce 1024 dimensional feature vector for each frame.

- Training Details

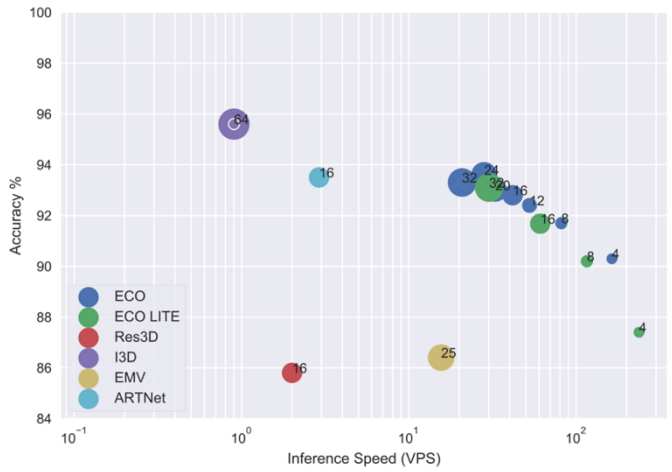
We split each video into N segments and randomly select one frame from each segment. This sampling provides robustness to variations and enables the network to fully exploit all frames. In addition, we re-size the input frames to 240 × 320 and employ fixed-corner cropping and scale jittering with horizontal flipping. Afterward, we run per-pixel mean subtraction and resize the cropped regions to 224 × 224.

The initial learning rate is 0.001 and decreases by a factor of 10 when validation error saturates for 4 epochs. We train the network with a momentum of 0.9, a weight decay of 0.0005, and mini-batches of size 32.

We finetune the above ECO/ECO Lite models on the new datasets. Due to the complexity of the dataset, we finetune the network for 25 epochs reducing the learning rate every 10 epochs by a factor of 10. For the rest, we finetune for 4k iterations and the learning rate drops by a factor of 10 as soon as the validation loss saturates.

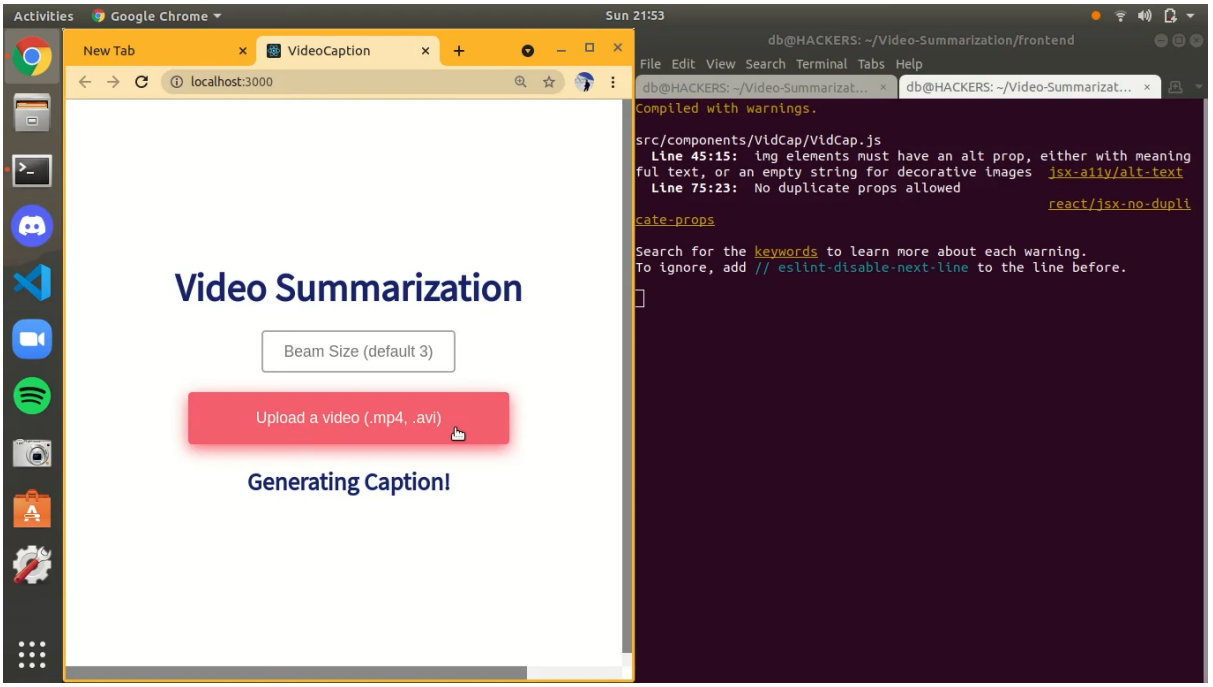


7) Experiments



Accuracy-runtime trade-off on UCF101 for various versions of ECO and other state-of-the-art approaches. ECO is much closer to the top right corner than any other approach.

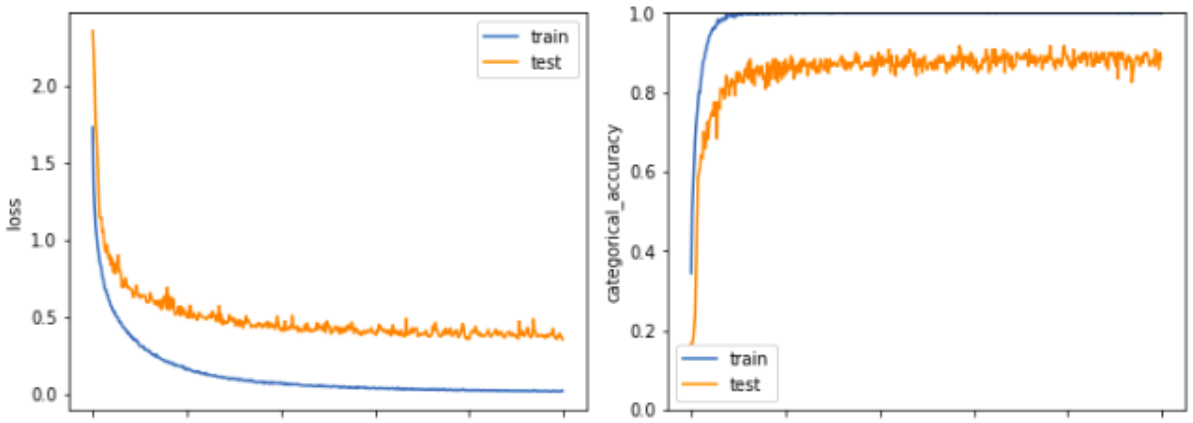
After having run a successful round of the initially implemented paper, we made front-end and back-end development for the same with his personal knowledge of full-stack development. And successfully implemented it.



The Github link can be accessed here.

8)Results

The accuracy obtained on training data is 100 %
The accuracy obtained on test data is 92.2 %



Model	Sampled Frames	Accuracy (%) UCF101	Kinetics
ECO	4	90.3	66.2
	8	91.7	67.8
	16	92.8	69.0
	32	93.3	67.8
ECO Lite	4	87.4	57.9
	8	90.2	–
	16	91.6	64.4
	32	93.1	–

Accuracy and runtime of ECO and ECO Lite for different numbers of sampled frames. The reported runtime is without considering I/O.

The above table compares the two architecture variants ECO and ECO Lite and evaluates the influence on the number of sampled frames N. As expected, the accuracy drops when sampling fewer frames, as the subsections get longer and important parts of the action can be missed. However, even with just 4 samples the accuracy of ECO is still much better than most approaches in literature, since ECO takes into account the relationship between these 2 instants in the video, even if they are far apart.

The advantages of the ECO architectures becomes even more prominent as we look at the accuracy-runtime trade-off. The ECO architectures yield the same accuracy as other approaches at much faster rates.

Previous works typically measure the speed of an approach in frames per second (fps). Our model with ECO runs at much fater fps (425 in our case, which varies in each case based on the GPU used). However, this does not reflect the time needed to process a whole video. This becomes relevant for methods like TSN and ours, which do not look at every frame of the video, and

motivates us to report videos per second (vps) to compare the speed of video understanding methods.

ECO can process videos at least an order of magnitude faster than the other approaches, making it an excellent architecture for video retrieval applications.

Method	UCF101 (%)
Res3D	85.8
TSN	87.7
EMV	86.4
I3D	95.6
ARTNet	93.5
$\text{ECO}_{Lite-4F}$	87.4
ECO_{4F}	90.3
ECO_{12F}	92.4
ECO_{20F}	93.0
ECO_{24F}	93.6

9)Work Distribution

Ankit : Wrote doc details of (intro and conclusion)literature review in PPT, wrote the literature review of ECO paper,network architecture of Conv2d, Conv3D layer and Resnet3D in network architecture and Eco-lite network architecture in pytorch.
Lochan : Contributed in Paper review and Doc, Slides in PPT, preprocessed the kinetic dataset with loading it and its pre-trained weights into the model and evaluate the result though inference.py file.
Utkarsh : Wrote Paper review, doc and slides preparation for mid eval, speaker in mid eval presentation, Added inception layer to ECO2D, Loading pre-trained weights of kinetic dataset into the model and evaluated on the network model, Extracted UCF dataset into frames of each class.
Vinamra : Wrote Paper review, implemented Eco-lite network architecture in pytorch, wrote data loader scripts for test and validation dataset, convert kinetics video data to image data(frames) using ffmpeg and also try to finetune model with ucf101 dataset.
Dev : Completed Project Poster Creation, documentation, was speaker for mid eval. Was entrusted with the prerequisite part of the project and helped in successful completion of it, uploaded the C3D Dataset, UCF101 and worked on Video Captioning and Real-Time Online Video Understanding.
https://github.com/xDB-9/Video-Captioning
This is the GitHub Repository where you'll find the working model of Video Captioning. Dev has implemented that as well while doing this project. Kindly have a look at it.

10)References

We referred to the following links of some research papers and articles for our overall work :

- https://arxiv.org/pdf/1804.09066v2.pdf
- https://www.kaggle.com/search?q=deep+learning
- https://www.kaggle.com/general/223029
- https://deepmind.com/research/open-source/kinetics