CS6910 : Deep Learning (for Computer Vision)

Vybhav Nath CA & Irfaan Arif
ME18B036,ME18B048

Course Project

Report on the "ECO: Efficient Convolutional Network for Online Video Understanding" paper

Introduction

In our paper, the authors reason that the state of the art networks in video understanding suffers from the following two major problems:

- Most networks rely on doing analysis on the video locally, hence loosing valuable information encoded in the long-term context of the video, for example actions that span several seconds.
- Information in a lot of the neighboring frames are largely redundant, and hence the processing of the whole video is not efficient and hampers fast video retrieval or online classification

In this paper, a new network architecture is introduced that takes into account long-term content and also 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. Also a sampling strategy is used to exploit the fact that the neighboring frames are largely redundant. This approach claims to achieve competitive performance across all datasets while being much faster than existing state-of-the-art methods.

We will be trying to verify these claims by running experiments and training models from scratch on the following datasets:

1. Kinetics 400
2. ActivityNet
3. Something-Something
4. UCF101
5. HMB51
6. Jester

The obtained validation/test results will be compared with the results published in the paper

Along with this, we will also be running various experiments based on activity recognition such as :

1. Effect of Video Blurring on the input
2. Applying a mask on each frame
3. Brightness Dimming
4. 1 channel Black White conversion
5. Applying Filters like 'sepia'
6. Random frame removal

Results of these experiments are reported and some examples will also be attached. We used the AWS instances p2.xlarge (Tesla K80) and g4dn.2xlarge (Tesla T4) to train and run our models.

Results of network training

For each of the datasets, we mainly focused on training the ECO-lite model, since we are going to evaluate our models based on action classification. The 3D architecture in ECO is optimized for learning relationships between the frames but it tends to waste capacity in case of simple short-term actions that can be recognized just from the static image content.

A sampling length of 16 was used and input modality of RGB only was used in the experiments. Also the results reported below are on the validation/test set of the datasets.

Dataset used	Np: of classes	%Test Accuracy	Reported Accuracy	% Difference
Kinetics 400	400	62.2	64.4	1.50
20BN Something-Something v1	174	39.6	42.2	6.16
UCF101	101	90.9	91.6	0.76
HMDB51	51	68.3	68.2	0.15
20BN Jester	27	85.3	-	-
ActivityNet	200	78.4	-	-

Table 1: Test accuracy for each dataset

Dataset used	Current SOTA Model	%Test Accuracy
20BN Something-Something v1	RGB-only 16f+8f	54.01
Kinetics 400	SlowFast, R101 + NL	79.8
ActivityNet	SMART	84.4
HMDB51	R2+1D-BERT	85.10
20BN Jester	MobileNet+NL+SlowFast	97.26
UCF101	R2+1D-BERT	98.69

Table 2: Current state of the art(SOTA) models

We utilized the same method as used in the paper to train our models. We train our networks using mini-batch SGD with Nesterov momentum and utilize dropout in each fully connected layer. An interesting point to note is that in the Caffe implementation the dropout is after the global pool layer but in the pytorch version, it's in the final layer.

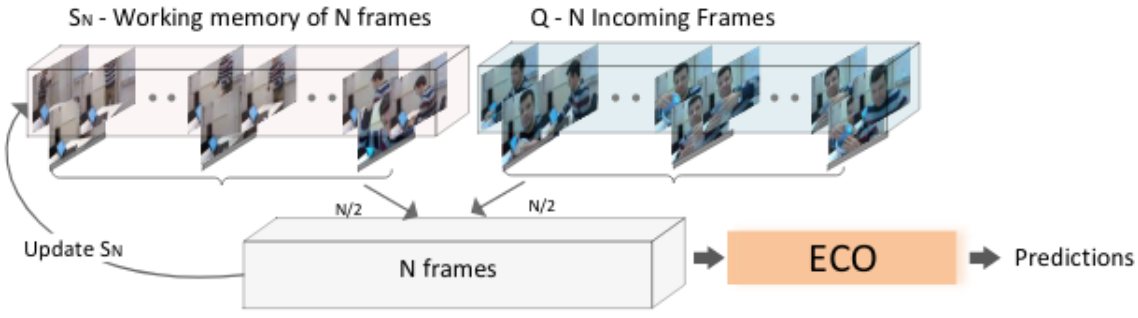


Figure 1: The sampling strategy use in online video understanding.

Network architecture

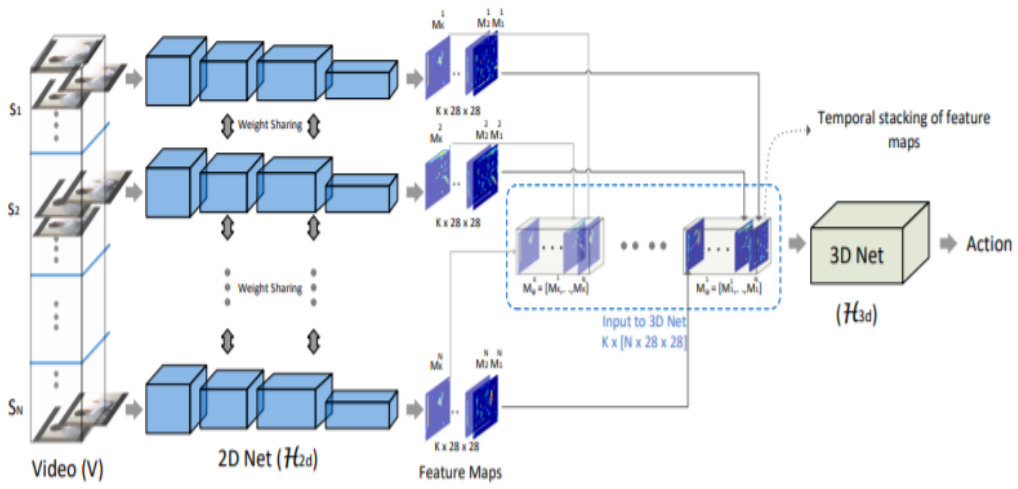


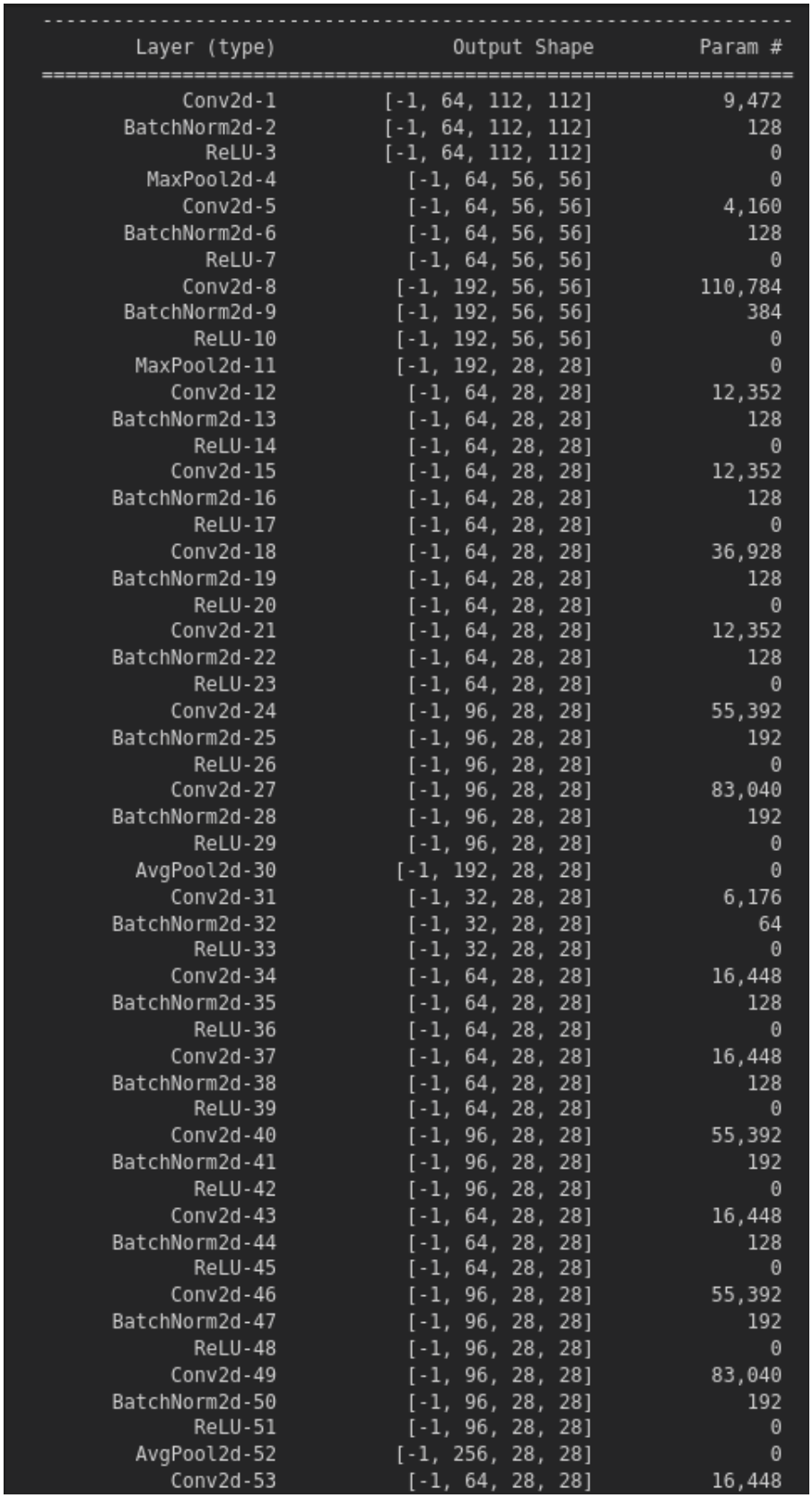
Figure 2: 'ECO-Net Architecture from the original paper'

Here is an architecture overview of ECO Lite. Each video is split into N subsections of equal size. From each subsection a single frames is randomly sampled. The samples are processed by a regular 2D convolutional network to yield a representation for each sampled frame. These representations are stacked and fed into a 3D convolutional network, which classifies the action, taking into account the temporal relationship. The 3D convolutional network used is mostly a pre-trained Resnet(mostly 18 or 34) on the Imagenet data set. Then the full network is trained and then the pre-trained weights are finetuned to get the optimum results.

Here we have attempted to tackle the problem of 'Video Action Recognition' using a pre-trained version of this model. This network can be modified with the help of LSTM's to solve the problem of 'Video Captioning'. Since video captioning is computationally very expensive, we just went ahead with the problem of action recognition

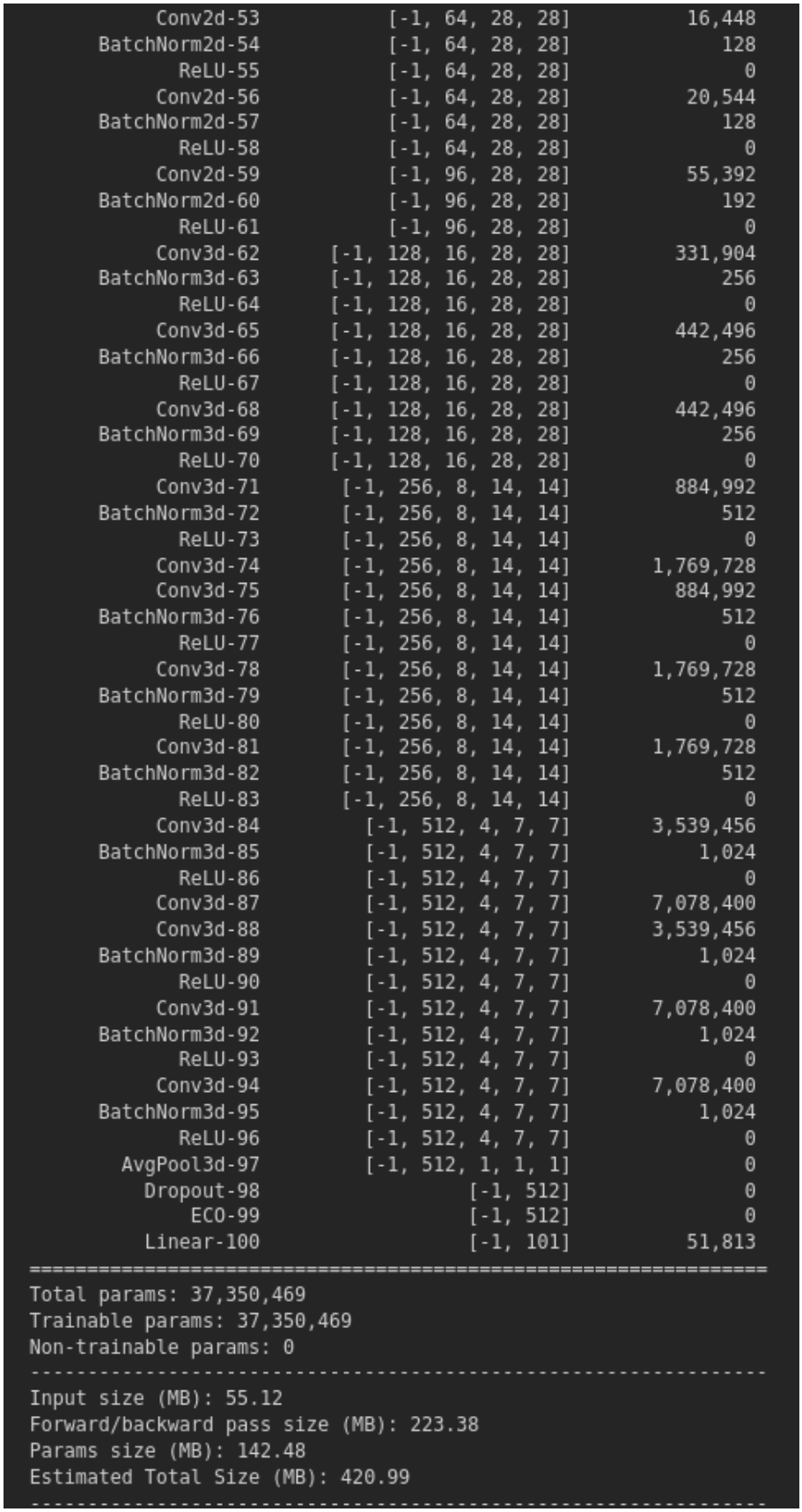
The pre-trained model that we used was trained on the 'Kinetics 400' data set for action recognition. The data set has 400 classes of common actions and many video corresponding to each class accounting to a total of 300k videos.Each class has at least 400 videos. Also the data set is not properly balanced. The complete model architecture with specifics has been attached as Figure's 4 and 5.

Figure 3: Rough overview of the architecture from the original paper. Also shows the case with a parallel 2D and 3D stream



Layer (type)	Output Shape	Param #
Conv2d-1	[-1, 64, 112, 112]	9,472
BatchNorm2d-2	[-1, 64, 112, 112]	128
ReLU-3	[-1, 64, 112, 112]	0
MaxPool2d-4	[-1, 64, 56, 56]	0
Conv2d-5	[-1, 64, 56, 56]	4,160
BatchNorm2d-6	[-1, 64, 56, 56]	128
ReLU-7	[-1, 64, 56, 56]	0
Conv2d-8	[-1, 192, 56, 56]	110,784
BatchNorm2d-9	[-1, 192, 56, 56]	384
ReLU-10	[-1, 192, 56, 56]	0
MaxPool2d-11	[-1, 192, 28, 28]	0
Conv2d-12	[-1, 64, 28, 28]	12,352
BatchNorm2d-13	[-1, 64, 28, 28]	128
ReLU-14	[-1, 64, 28, 28]	0
Conv2d-15	[-1, 64, 28, 28]	12,352
BatchNorm2d-16	[-1, 64, 28, 28]	128
ReLU-17	[-1, 64, 28, 28]	0
Conv2d-18	[-1, 64, 28, 28]	36,928
BatchNorm2d-19	[-1, 64, 28, 28]	128
ReLU-20	[-1, 64, 28, 28]	0
Conv2d-21	[-1, 64, 28, 28]	12,352
BatchNorm2d-22	[-1, 64, 28, 28]	128
ReLU-23	[-1, 64, 28, 28]	0
Conv2d-24	[-1, 96, 28, 28]	55,392
BatchNorm2d-25	[-1, 96, 28, 28]	192
ReLU-26	[-1, 96, 28, 28]	0
Conv2d-27	[-1, 96, 28, 28]	83,040
BatchNorm2d-28	[-1, 96, 28, 28]	192
ReLU-29	[-1, 96, 28, 28]	0
AvgPool2d-30	[-1, 192, 28, 28]	0
Conv2d-31	[-1, 32, 28, 28]	6,176
BatchNorm2d-32	[-1, 32, 28, 28]	64
ReLU-33	[-1, 32, 28, 28]	0
Conv2d-34	[-1, 64, 28, 28]	16,448
BatchNorm2d-35	[-1, 64, 28, 28]	128
ReLU-36	[-1, 64, 28, 28]	0
Conv2d-37	[-1, 64, 28, 28]	16,448
BatchNorm2d-38	[-1, 64, 28, 28]	128
ReLU-39	[-1, 64, 28, 28]	0
Conv2d-40	[-1, 96, 28, 28]	55,392
BatchNorm2d-41	[-1, 96, 28, 28]	192
ReLU-42	[-1, 96, 28, 28]	0
Conv2d-43	[-1, 64, 28, 28]	16,448
BatchNorm2d-44	[-1, 64, 28, 28]	128
ReLU-45	[-1, 64, 28, 28]	0
Conv2d-46	[-1, 96, 28, 28]	55,392
BatchNorm2d-47	[-1, 96, 28, 28]	192
ReLU-48	[-1, 96, 28, 28]	0
Conv2d-49	[-1, 96, 28, 28]	83,040
BatchNorm2d-50	[-1, 96, 28, 28]	192
ReLU-51	[-1, 96, 28, 28]	0
AvgPool2d-52	[-1, 256, 28, 28]	0
Conv2d-53	[-1, 64, 28, 28]	16,448

Figure 4: Complete network architecture pt1.



```
           Conv2d-53          [-1, 64, 28, 28]          16,448
      BatchNorm2d-54          [-1, 64, 28, 28]             128
             ReLU-55          [-1, 64, 28, 28]               0
           Conv2d-56          [-1, 64, 28, 28]          20,544
      BatchNorm2d-57          [-1, 64, 28, 28]             128
             ReLU-58          [-1, 64, 28, 28]               0
           Conv2d-59          [-1, 96, 28, 28]          55,392
      BatchNorm2d-60          [-1, 96, 28, 28]             192
             ReLU-61          [-1, 96, 28, 28]               0
           Conv3d-62      [-1, 128, 16, 28, 28]         331,904
      BatchNorm3d-63      [-1, 128, 16, 28, 28]             256
             ReLU-64      [-1, 128, 16, 28, 28]               0
           Conv3d-65      [-1, 128, 16, 28, 28]         442,496
      BatchNorm3d-66      [-1, 128, 16, 28, 28]             256
             ReLU-67      [-1, 128, 16, 28, 28]               0
           Conv3d-68      [-1, 128, 16, 28, 28]         442,496
      BatchNorm3d-69      [-1, 128, 16, 28, 28]             256
             ReLU-70      [-1, 128, 16, 28, 28]               0
           Conv3d-71       [-1, 256, 8, 14, 14]         884,992
      BatchNorm3d-72       [-1, 256, 8, 14, 14]             512
             ReLU-73       [-1, 256, 8, 14, 14]               0
           Conv3d-74       [-1, 256, 8, 14, 14]       1,769,728
           Conv3d-75       [-1, 256, 8, 14, 14]         884,992
      BatchNorm3d-76       [-1, 256, 8, 14, 14]             512
             ReLU-77       [-1, 256, 8, 14, 14]               0
           Conv3d-78       [-1, 256, 8, 14, 14]       1,769,728
      BatchNorm3d-79       [-1, 256, 8, 14, 14]             512
             ReLU-80       [-1, 256, 8, 14, 14]               0
           Conv3d-81       [-1, 256, 8, 14, 14]       1,769,728
      BatchNorm3d-82       [-1, 256, 8, 14, 14]             512
             ReLU-83       [-1, 256, 8, 14, 14]               0
           Conv3d-84         [-1, 512, 4, 7, 7]       3,539,456
      BatchNorm3d-85         [-1, 512, 4, 7, 7]           1,024
             ReLU-86         [-1, 512, 4, 7, 7]               0
           Conv3d-87         [-1, 512, 4, 7, 7]       7,078,400
           Conv3d-88         [-1, 512, 4, 7, 7]       3,539,456
      BatchNorm3d-89         [-1, 512, 4, 7, 7]           1,024
             ReLU-90         [-1, 512, 4, 7, 7]               0
           Conv3d-91         [-1, 512, 4, 7, 7]       7,078,400
      BatchNorm3d-92         [-1, 512, 4, 7, 7]           1,024
             ReLU-93         [-1, 512, 4, 7, 7]               0
           Conv3d-94         [-1, 512, 4, 7, 7]       7,078,400
      BatchNorm3d-95         [-1, 512, 4, 7, 7]           1,024
             ReLU-96         [-1, 512, 4, 7, 7]               0
        AvgPool3d-97         [-1, 512, 1, 1, 1]               0
          Dropout-98                  [-1, 512]               0
              ECO-99                  [-1, 512]               0
         Linear-100                  [-1, 101]          51,813
================================================================
Total params: 37,350,469
Trainable params: 37,350,469
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 55.12
Forward/backward pass size (MB): 223.38
Params size (MB): 142.48
Estimated Total Size (MB): 420.99
----------------------------------------------------------------
```

Figure 5: Complete network architecture pt2.

Random frame removal

We are also gonna be randomly removing some frames of the videos to see what effect it causes. We are going to be using the mdel which was trained on the UCF 101 dataset, and for our experiments we are using the tiny UCF 101 dataset which has one video for each of the 101 classes. We are going to be removing a random percentage of frames of the video, the results are as follows

% Frames removed	Accuracy
Baseline (0)	100/101
40	100/101
50	98/101
60	97/101
70	92/101
80	91/101

Table 3: Test accuracy for each dataset

Hence we can conclude that due to our sampling strategy the network is not completely reliant on the local context of every frame, and has learnt to undertand long term contextual information also.

Figure 6: True Label: SkyDiving

Figure 7: True Label: Skiing

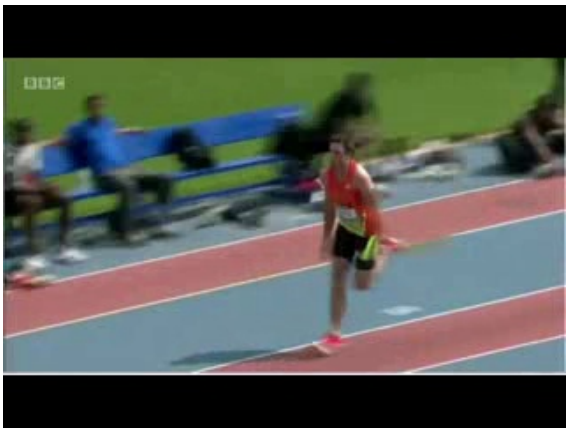


Figure 8: True Label: Long jump

Figure 9: True Label: Basketball dunk

Figures 6 and 7 show two of the many actions which were always correctly predicted, and Figures 8 and 9 show two classes which were mis-classified at 60% and 70% frames removed respectively.

Test

Here we tested the model trained on 'Kinetics 400' on a few videos which we obtained from google based on the classes given in the dataset.

True Class 'Counting Money'

Figure 10: 'Counting Money'

True Class 'Brush Painting'

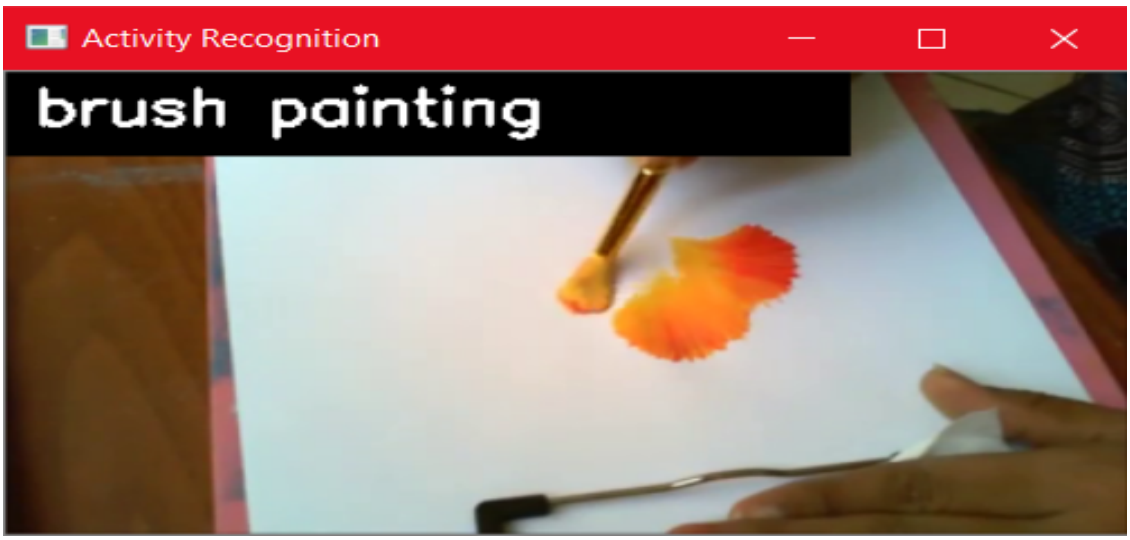


Figure 11: 'Brush Painting'

True Class 'Clapping/Applauding'

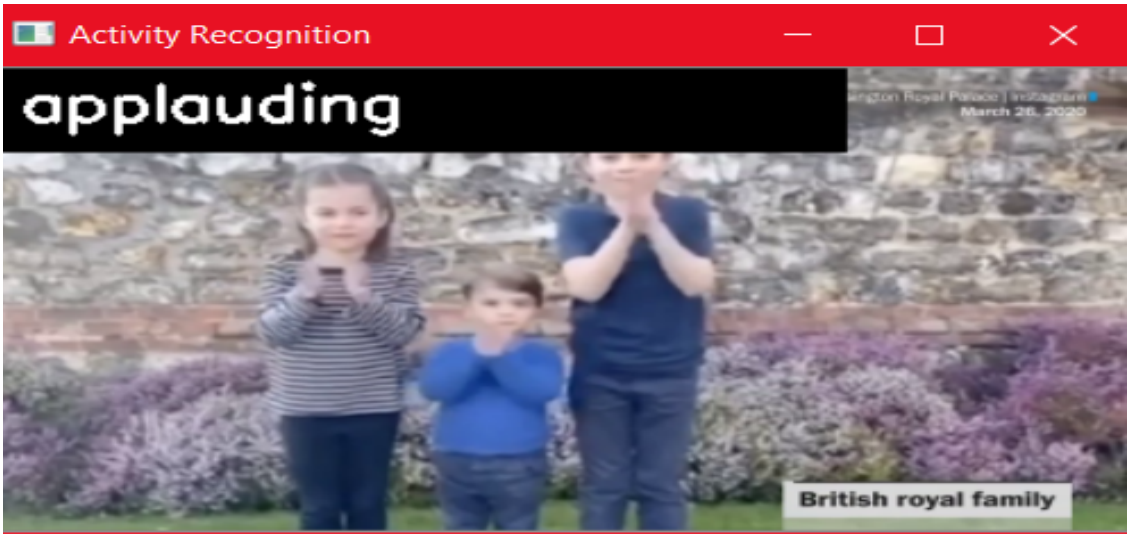


Figure 12: 'Clapping'

True Class 'Making Pizza'

Figure 13: 'Making Pizza'

Experiments

Video Blur

We applied a blur of value 5 on the videos of money counting and brush painting to see the effect of this blur on our network's accuracy. Here are the results:

True Class: 'Counting Money'

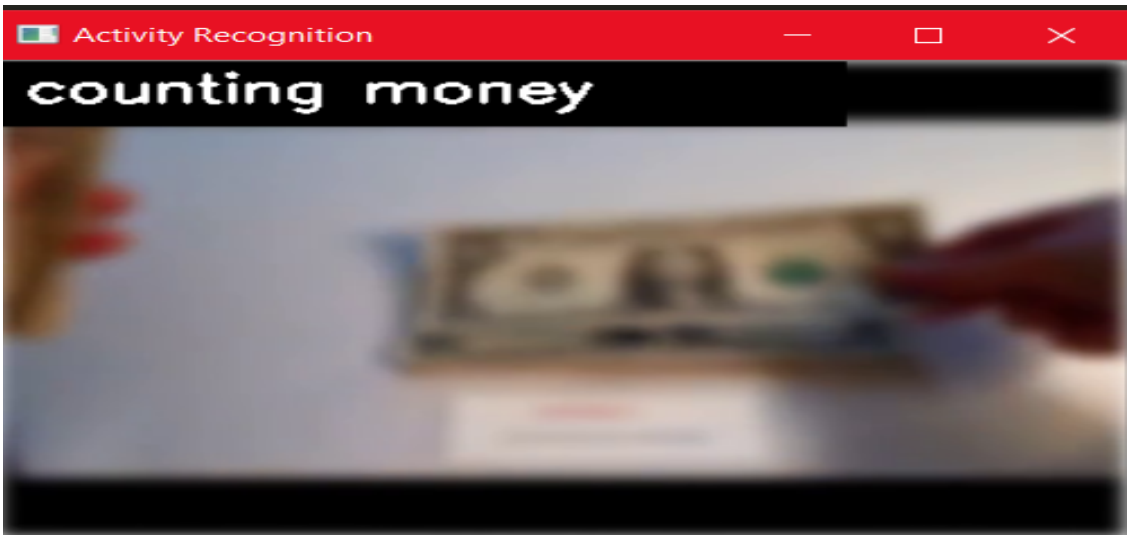


Figure 14: Blur 1

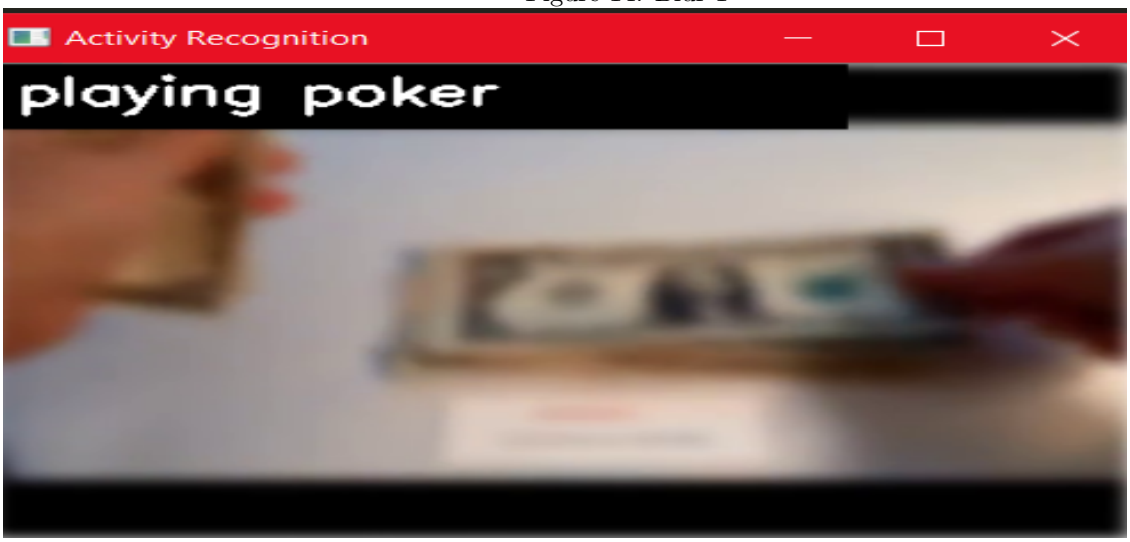


Figure 15: Blur 2

The network is getting confused between the true class and another class called 'Playing poker'

True Class: 'Brush Painting'

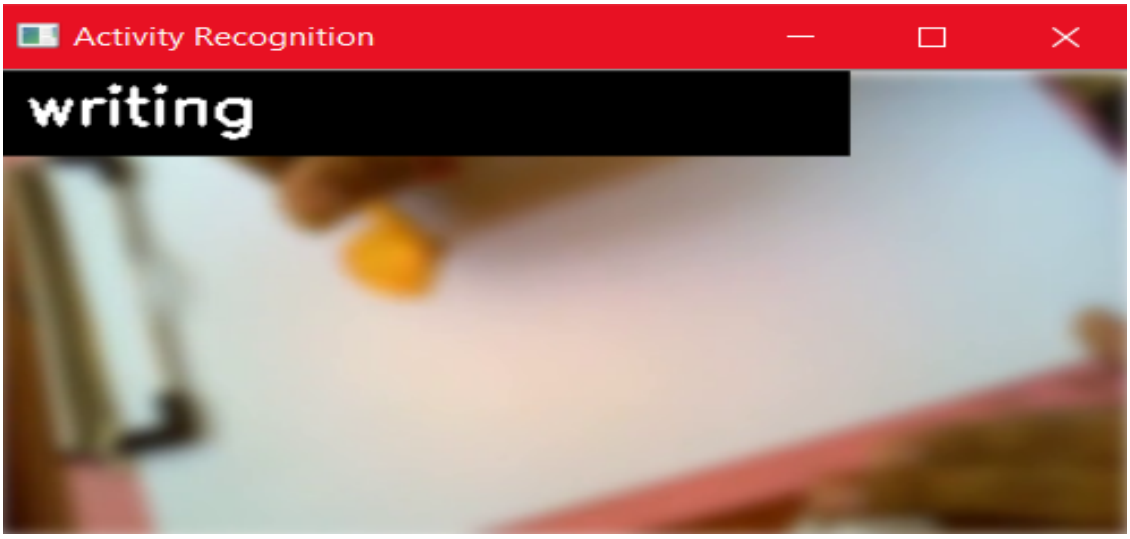


Figure 16: Blur 1

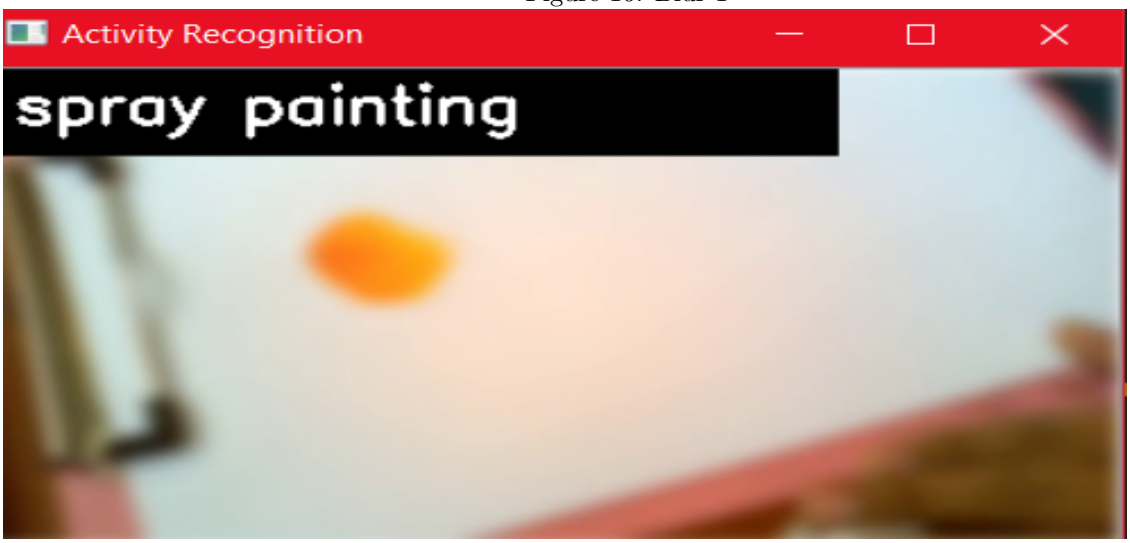


Figure 17: Blur 2

Figure 18: The predicted classes for the given true class

The network is getting confused between the classes 'writing' and another class called 'spray painting' which is closer to brush painting.

Video Masking

We applied a mask of size 100x100 on the full video and performed the prediction task. Here are the results:

True Class: 'Counting Money'

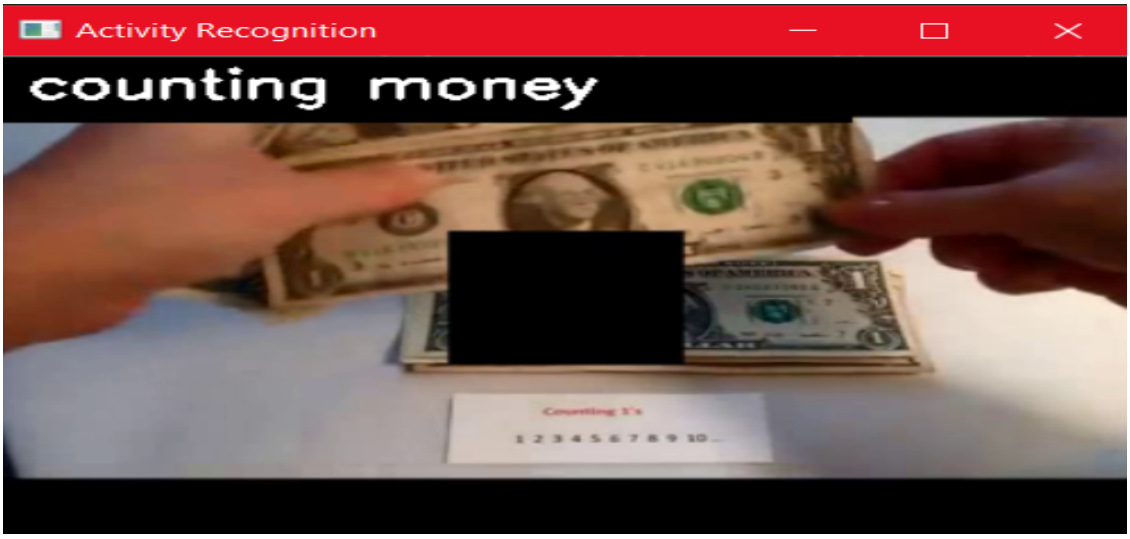


Figure 19: Mask 1

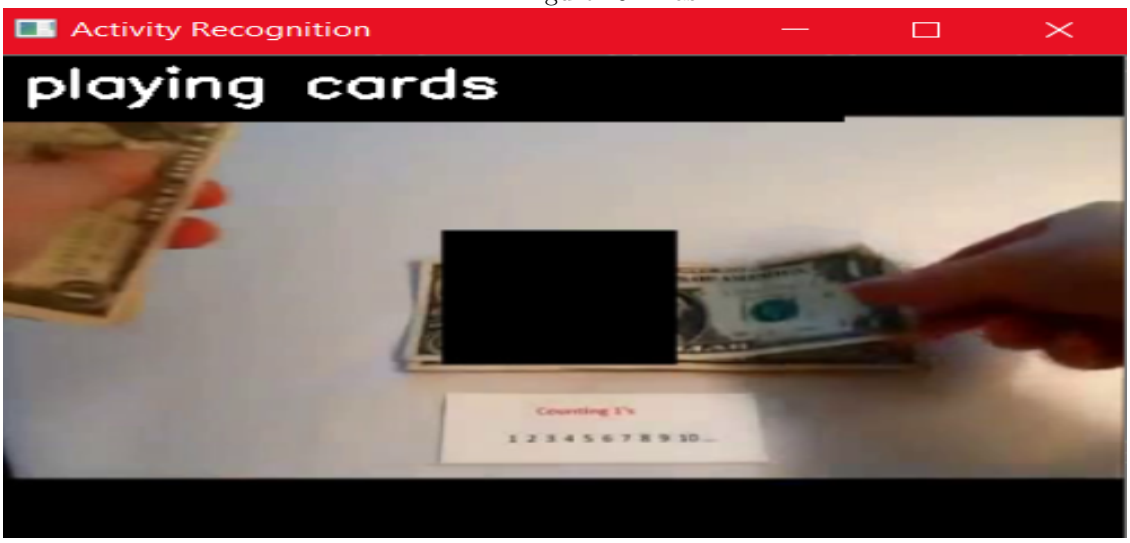


Figure 20: Mask 2

The network is getting confused between the true class, a new class called 'Unboxing' and the class 'Playing Poker'.

Brightness Dimming

We applied a brightness of -75 on the full video and performed the prediction task. Here are the results:

True Class: 'Counting Money'

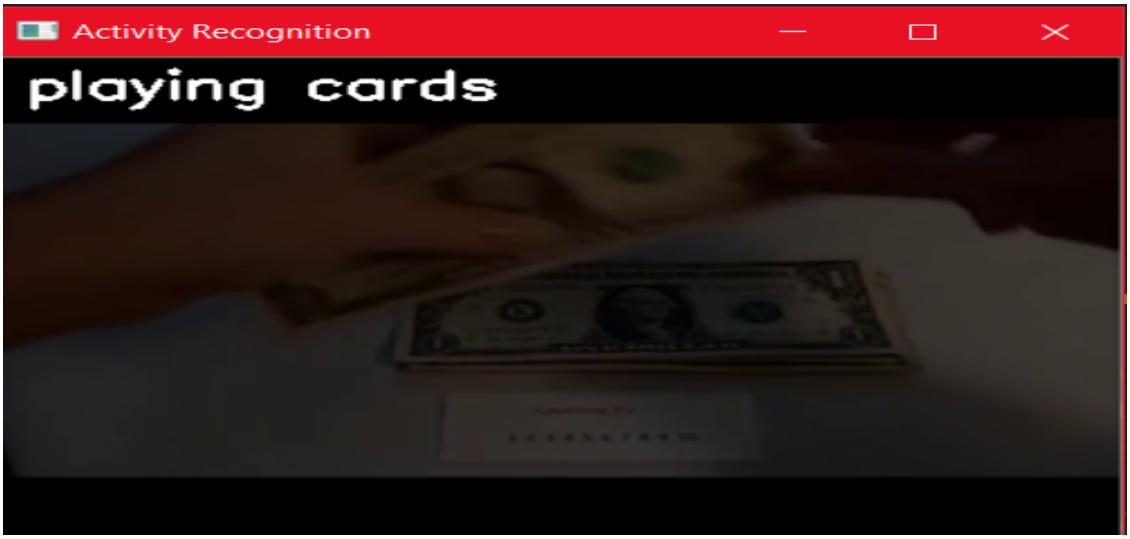


Figure 21: Brightness 1

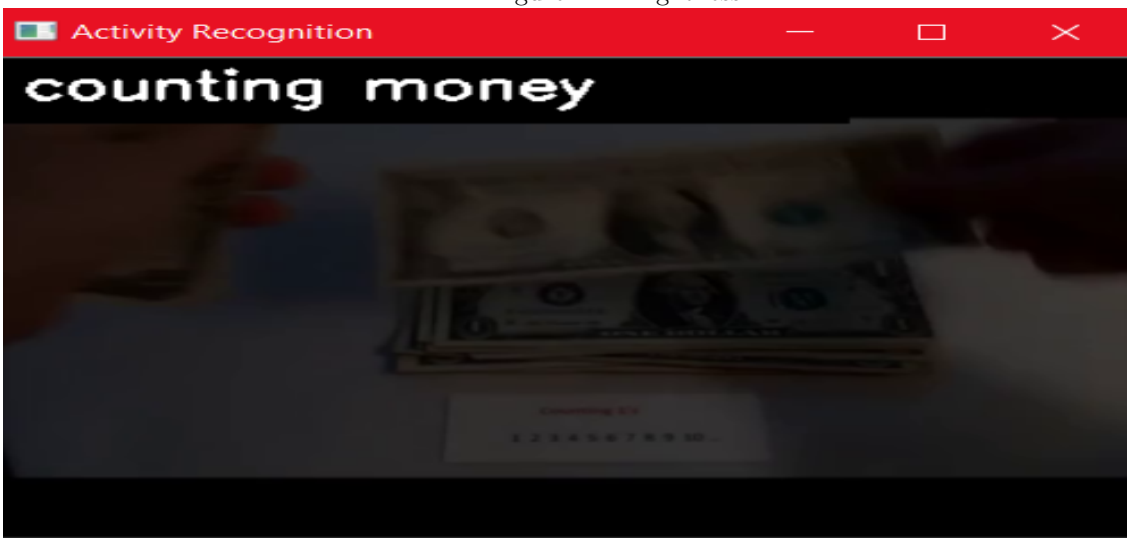


Figure 22: Brightness 2

The network is getting confused between the true class and another class called 'Playing poker' which it mostly predicted

True Class: 'Brush Painting'

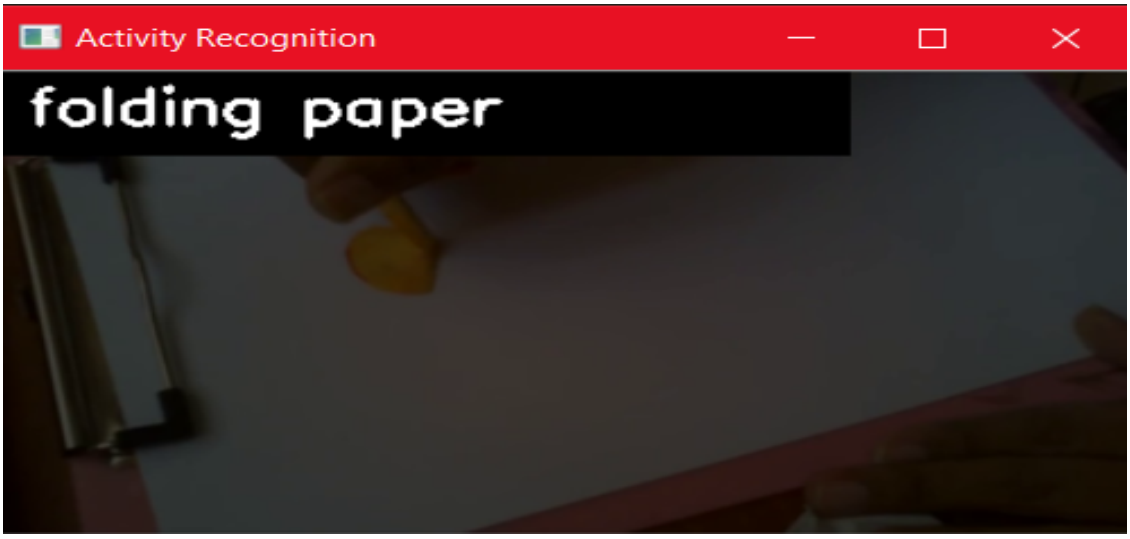


Figure 23: Brightness 1

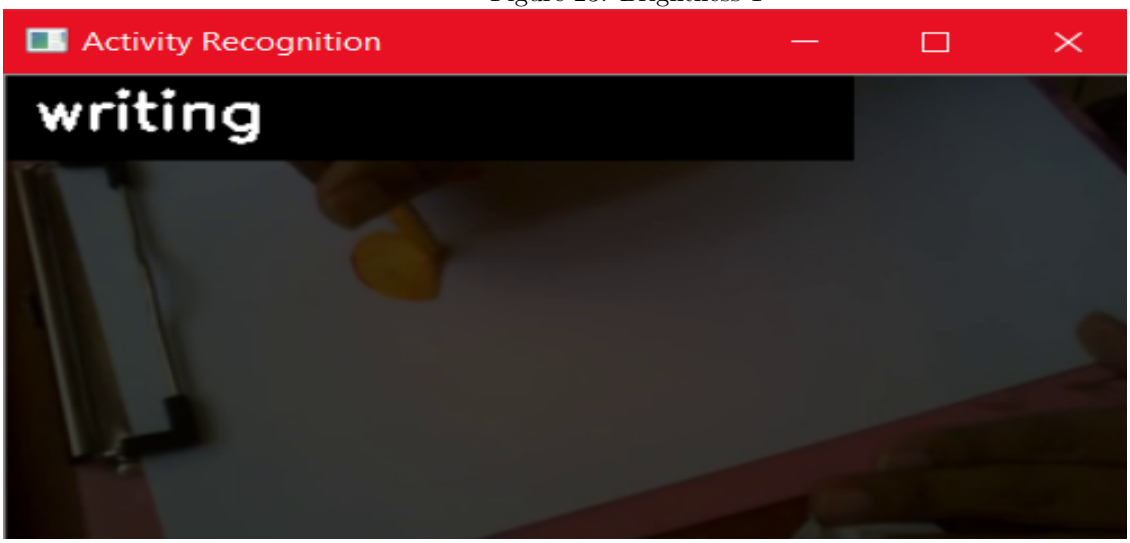


Figure 24: Brightness 2

Here the network is not able to properly identify the true class and instead predicts it to be 'Folding Paper' and 'Writing'. It predicted the class 'Spray painting' for 2 frames.

Black White Conversion

We converted the video to black and white and tried predicting. Here are the results:

True Class: 'Clapping/Applauding'

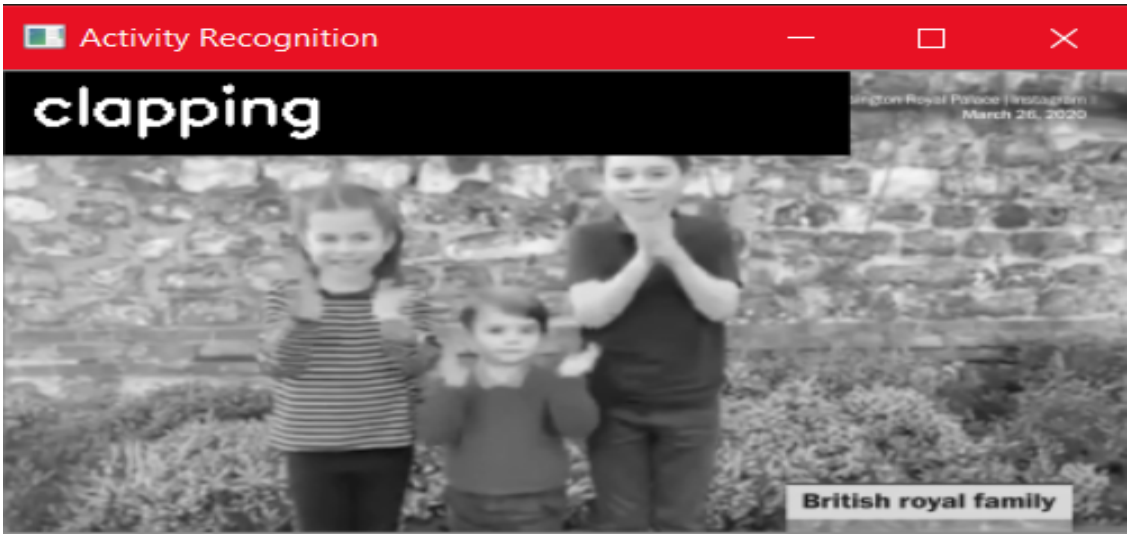


Figure 25: B W 1

Figure 26: B W 2

Here also the network is getting confused between the classes 'Clapping' and 'Reading Newspaper'. The Black and white image along with the folded hands seem to give an impression that they are reading a newspaper.

Applying Filter

Here we applied a very common filter in the video industry known as 'Sepia'. Then we tried predicting on the new video.

True Class: 'Clapping/Applauding'

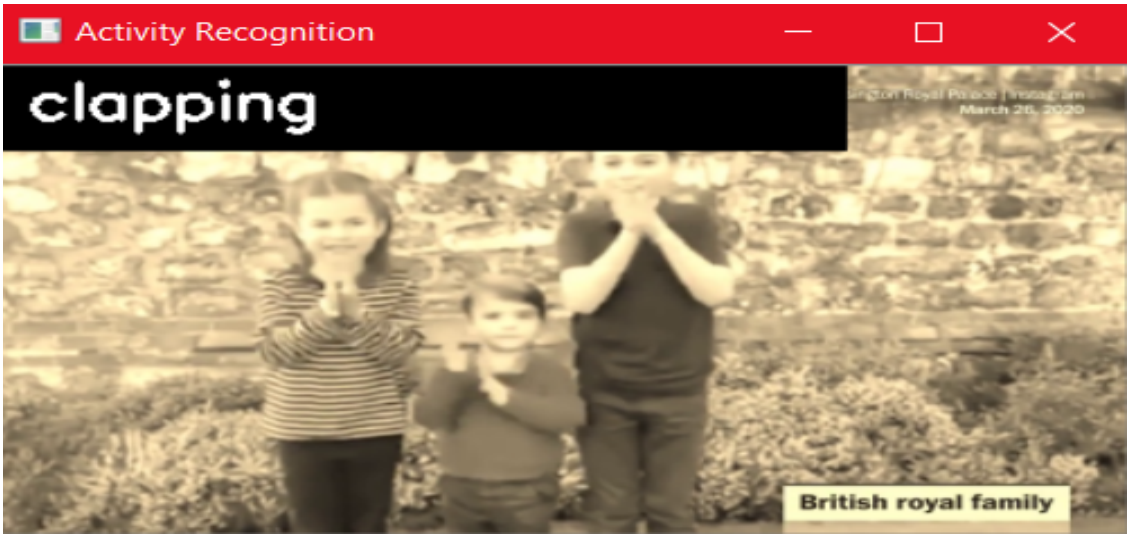


Figure 27: Sepia 1

Figure 28: Sepia 2

Here also the network is getting confused between the classes 'Clapping' and 'Contact Juggling'.

Conclusion

In this report we have performed a lot of experiments on the trained model and find what all affects the performance of our network. Our model has a hard time predicting the true class when the video was blurred and when the brightness of the video was dimmed.Even though it is able to predict the true class for some frames, it cannot continue the same for all frames and gets confused. Also these videos are not from the data set. We obtained these video from Youtube and other sources. So slight accuracy issues can be expected. Some

References

1. Efficient Convolutional Network for Online Video Understanding ECO-Net
2. Brush Painting
3. Clapping/Applauding
4. Counting Money
5. Making Pizza
6. Activity recognition benchmarks