Assignment 2
ELL888 : Advanced Machine Learning

Siddharth Khera
Indian Institute of Technology, Delhi
2015MT60567
mt6150567@iitd.ac.in

Kartikeya Sharma
Indian Institute of Technology, Delhi
2015CS10234
cs1150234@iitd.ac.in

Navreet Kaur
Indian Institute of Technology, Delhi
2015TT10917
tt1150917@iitd.ac.in

I. Problem Statement

Lots of good quality educational videos are available on platforms like YouTube and NPTEL but students are not able to use them optimally since videos are not indexed properly. This assignment tries to deal with this problem. The aim of this assignment is to extract out those frame of a video which captures the full written content on a sheet of paper. Suppose the professor writes 7 sheets in one lecture, if we could get the frames capturing those 7 filled sheets, we will have almost all the information of what professor taught in that lecture in just 7 images. One hour video can be summarised using 7 images (Call these 7 frames as 'key-frames').

II. Methods and Experiments

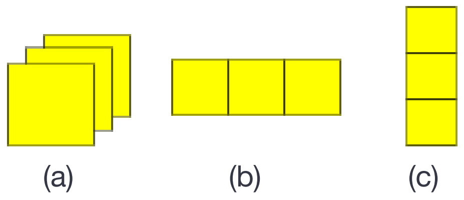
A. Input Format

We tried different methods of feeding the input to the neural network. We observed that the label 1 was occuring in the training set in the last frame before a transition of the scene, i.e., just before the new slide is presented. We tried to capture this in the input format itself by concatenating frames to the left and right side of the target frame. We tried different ways of concatenating these frames as follows:

1. *Concatenation of frames - channel wise*
   Frames were concatenated in the 3rd dimension after converting each to grayscale. For example, for a window size of 3, the pervious, current and next frame, stacked on top of each other (as shown in Figure 1(a)) will form one input sample.

2. *Concatenation of frames - side by side(different channels):*
   Frames were concatenated in the 2nd dimension, one after the other in the same plane. For example, for a window size of 3 the pervious, current and next frame(each of size AxB), joined together side by side will result into an input image of size Ax3B (as shown in Figure 1(b))

3. *Concatenation of frames - vertically:*
   Frames were concatenated in the 2nd dimension, one after the other in the same plane, vertically. For example, for a window size of 3 the pervious, current and next frame(each of size AxB), joined together side by side vertically will result into an input image of size 3AxB (as shown in Figure 1(c))


(a) (b) (c)
Figure 1: Concatenation of frames

4. *Features from intermediate layer of pre-trained VGG19 as inputs:*
   In this method, each image was passed through 3 convolution blocks of the VGG19 network pre-trained on the ImageNet and the

output from the intermediate layer (block3_pool, see Figure 4) was used as features for an image instead of the pixel intensities of the image itself. We then concatenated these features in a similar manner as described in the last 3 points and used them as inputs.

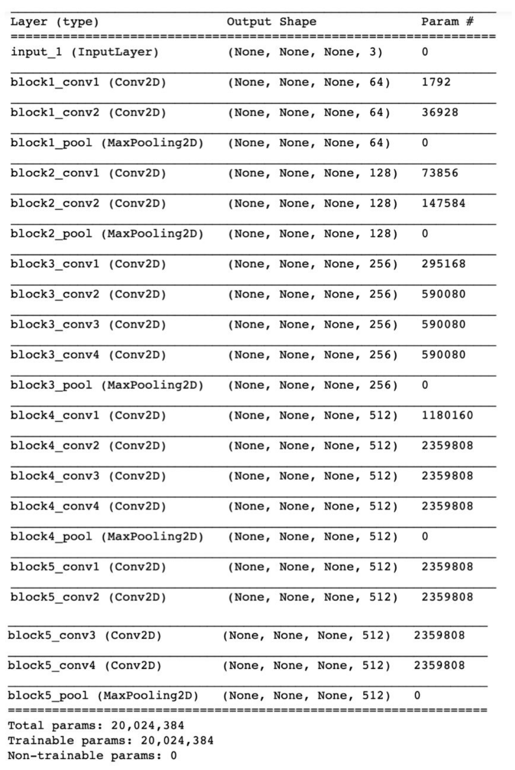
5. *Concatenation of Pixel Intensities with Computer Vision based features*

    a. *SAD: Sum of absolute differences -*
    Sum of absolute differences The two consecutive frames are compared pixel by pixel, summing up the absolute values of the differences of each two corresponding pixels.

    b. *HD: Histogram differences -*
    HD computes the difference between the histograms of two consecutive frames

    c. *Edge change ratio (ECR) -*
    The ECR attempts to compare the actual content of two frames. It transforms both frames to edge pictures, i. e. it extracts the probable outlines of objects within the pictures

6. *Labeling 'n' frames left of gold frame as 1:*
In this technique, we change the labels of the 'n' number of frames just left of the target frame from 0 to 1



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         (None, None, None, 3)     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, None, None, 64)    1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, None, None, 64)    36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, None, None, 64)    0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, None, None, 128)   73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, None, None, 128)   147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, None, None, 128)   0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, None, None, 256)   295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_conv4 (Conv2D)        (None, None, None, 256)   590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, None, None, 256)   0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, None, None, 512)   1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_conv4 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, None, None, 512)   0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_conv4 (Conv2D)        (None, None, None, 512)   2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, None, None, 512)   0
=================================================================
Total params: 20,024,384
Trainable params: 20,024,384
Non-trainable params: 0
```

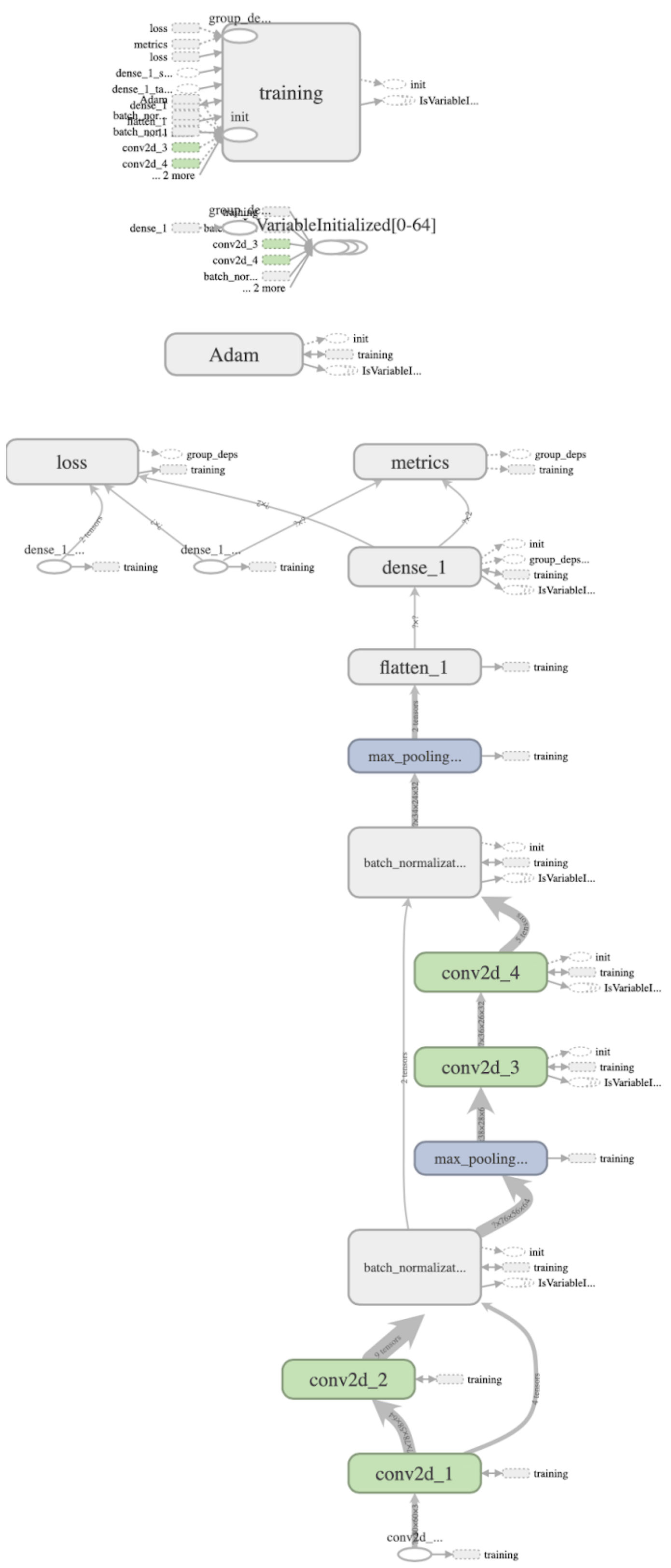
Figure 2 - VGG19 architecture

We observed that the training was very slow for the actual image size i.e. 240x360, so in the preprocessing step, we converted all the images to grayscale and resized all the frames to have a resolution of 60x80, except for the 4th case(VGG19 features), since (i) the model architecture assumes that the input image has 3 channels, and (ii) output after *block4_pool* in VGG19 already has a very small size.

B. Architecture

We tried several architecture tweaks like changing the order of layers, number of convolutions, types of pooling, use of regularizers like dropout and batch normalisation and here we report the three main approaches which gave the best results. The architecture of the best model is shown in Figure 3.

1. 2 Convolution layers without Batch Normalisation, without Dropout
2. 2 Convolution layers with Batch Normalisation, without Dropout
3. More convolutions help - 4 Convolution layers with Batch Normalisation



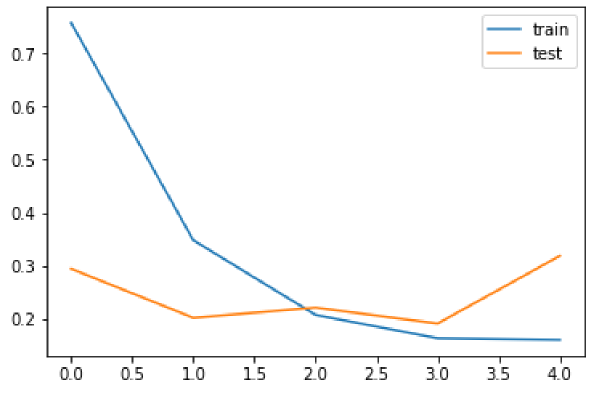
D. Hyperparameters

Due to the small data size, we were also able to experiment with various hyperparameters, like size of kernels, number of filters in each layer, dropout rate etc.

III. Observations

<u>*Early Stopping*</u>:

While doing early stopping, we observed that while the recall of the validation set is more if we stop training before the validation loss starts increasing, but the precision is still low. Therefore, we a make a trade-off between precision and recall and continue training a few epochs after the epoch at which validation loss starts rising. This ensures that we have good enough values of both the precision and the recall.



<u>*Window Size:*</u>

Increasing the window size worsens the performance. This is probably because the dependence of the frame's label is more on the immediate previous and immediate future frame rather than the history or future frames spanning a greater length, because, if the left and right frame both are similar to the current frame, then label is 0, otherwise, if the left is more similar and right is different/ not very similar, the label is 1, and in both of these cases, it doesn't really matter

<u>*Upsampling:*</u>

Since there was quite a disbalance between the classes, with frames in the positive class belonging to only 4% of the data, we did upsampling of the label 1 frames, with different ratio of number of positive to negative class instances. This ratio (scale ones to zeros) is another hyperparameter.

In the subsequent section, we report results corresponding to the best architecture and hyperparameters with different scale ones to zeros.

Precision, Recall, F1 Scores for different 1-0 ratios and window size of 3

scale ones to zero = 1

	precision	recall	f1-score	support
0	1.00	0.98	0.99	2798
1	0.67	0.90	0.77	109

scale ones to zero = 0.5

	precision	recall	f1-score	support
0	0.99	0.99	0.99	2798
1	0.87	0.86	0.87	109

scale ones to zero = 0.25

	precision	recall	f1-score	support
0	0.99	0.98	0.99	2798
1	0.65	0.82	0.72	109

scale ones to zero = 0.5 (*with Canny edge*)

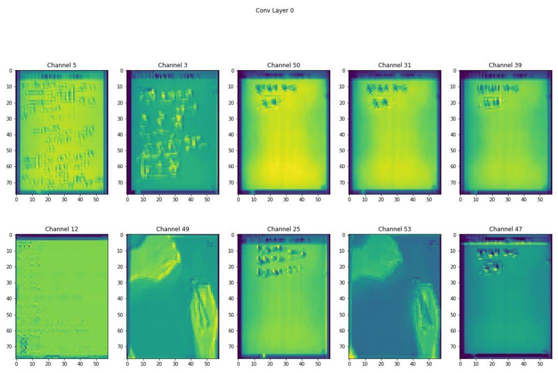
	precision	recall	f1-score	support
0	0.99	0.99	0.99	2798
1	0.84	0.86	0.85	109

scale ones to zero = 0.5 (*with Horizontally stacked 3 images*)

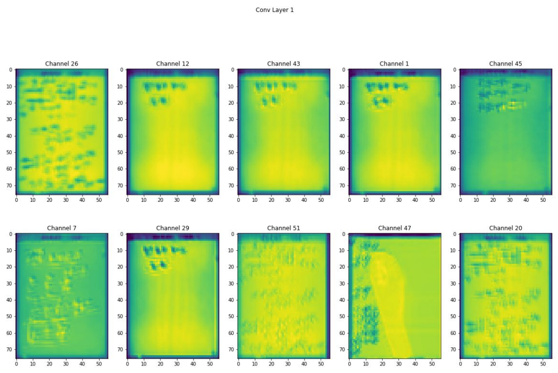
	precision	recall	f1-score	support
0	0.98	0.96	0.97	2798
1	0.35	0.54	0.42	109

The following images show a visualisation of the maximum activated channels per convolutional layer across all examples. These were generated by calculating the activations of the conv layers for each example and summing it across each channel. Then the maximum sum channels were selected and visualized.

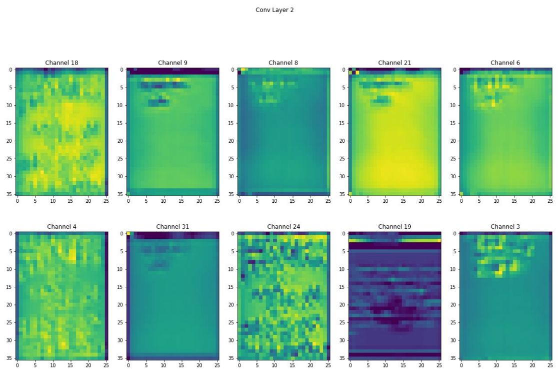
This helps us see what sort of features the network has learnt. We can observe that in the initial layers, the network learns simple features, such as shape of the professor's hands as they are placed on the desk, and the shape of the text on paper etc. As we proceed to deeper layers, it becomes hard to understand what concepts the network is learning. It is also important to note that while creating these visualizations, we observed many of the channels to have a completely zero activation (all activations in that channel were zero). This is due to the ReLU neurons dying, which happens quite often when using this activation function.



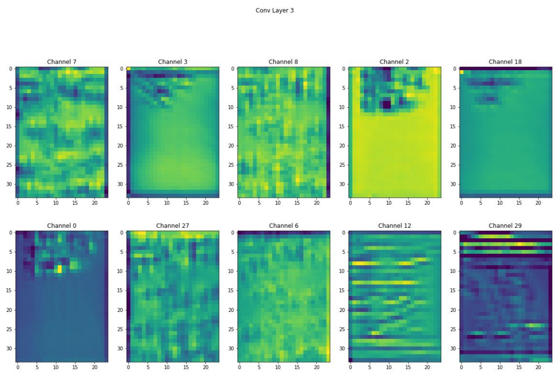
Max activation layer 0



Max activation layer 1

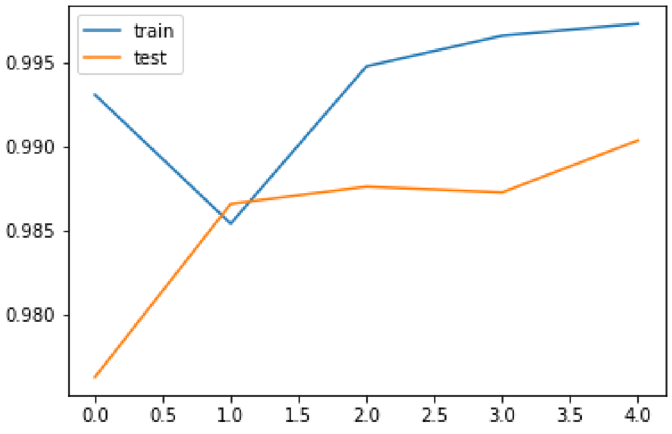


Max activation layer 2

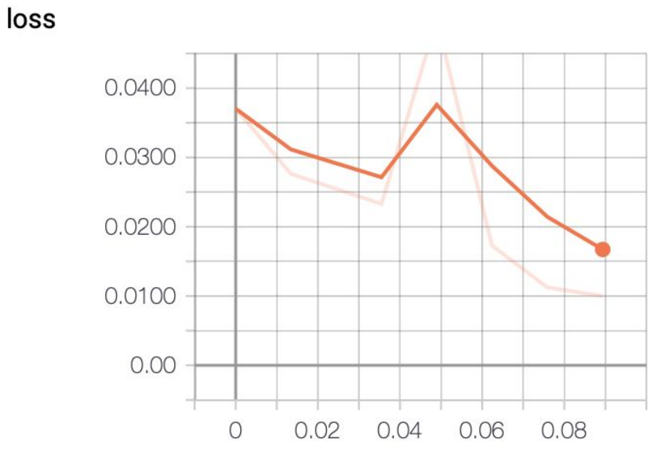


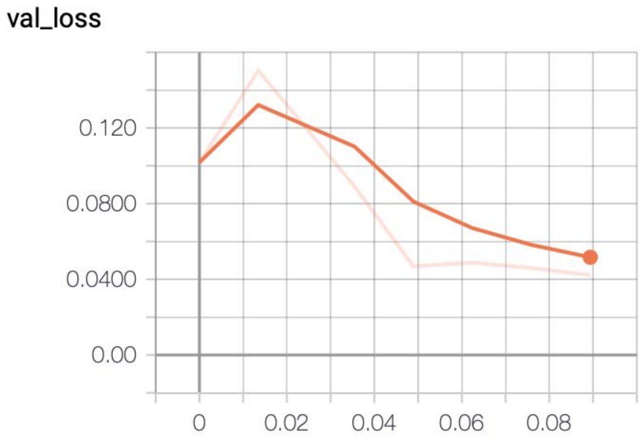
Max activation layer 3

The following graph shows the training and testing accuracy with respect to each epoch. This is when we stop the training before a particular epoch (found by doing early stopping).



Training vs. Validation Accuracy





Training vs. Validation Loss

Concatenating CNN dense layer with engineered features:

Engineered Features used :
The following features are known to work well for shot transition detection:

SAD : Sum of absolute differences
The two consecutive frames are compared pixel by pixel, summing up the absolute values of the differences of each two corresponding pixels.

HD : Histogram differences
HD computes the difference between the histograms of two consecutive frames

Edge change ratio (ECR). The ECR attempts to compare the actual content of two frames. It transforms both frames to edge pictures, i. e. it extracts the probable outlines of objects within the pictures

VGG as input features

After the flatten layer of CNN, a dense layer of 10 neurons is added. The 3 scalar features are concatenated to this layer and passed through another dense layer of size 10, followed by the softmax.

Effect of Dropout: It was noticed that using dropout reduced the learning ability of the model considerably. It is probably because the dataset is so small, that using dropout provides too strong a regularization effect for any useful learning. Instead, we chose to use batch normalization for regularization.