**EXPERIMENT DOCUMENTATION**

For the first experiment we stick to a very basic model using only 3D-convolution and dense layers. The input size is taken as (120, 120, 3). The reason we start with (120,120,3) is because a smaller input results in a lower number of parameters (obtaining the smallest model being one of the main objective. Also we take only 10 images/video skipping 2 images between each selection. This is done as in each video we see that the intermediate frames are mostly similar, so we try to lay more importance on the tails.

**Experiment 1:**

1. Architecture used: 3d-conv 4 layers followed by single dense layer, with max-pooling: (2,2,2) in first two convolution layers and (1,2,2) in next 2
2. dropout between final dense layers: 0.8
3. filter size = (3,3,3) for all layers
4. number of parameters = 58917
5. filter number : 32,32, 16,16
6. activation function : relu in conv3d, softmax in output
7. For each video 10 frames are used in both train and validation (skipping 2 between each frame)
8. Epoch = 50
9. batch size = 32
10. Image augmentation = NO
11. reduction\_on\_plateau : factor = 0.2, min\_lr= 0.001
12. optimiser : adam starting learning rate = 0.001
13. loss : categorical cross-entropy

**Results**

***final train accuracy***: 92.04 %; ***final train loss***: .2047

***final validation accuracy***: 77.78%; ***final validation loss***: .7826

**Analysis**

We see that the model is overfitting. Possible steps to take:

1. Increase the number of images per video. This will increase the training data and hence will reduce overfitting.
2. Introduce regularization, inform of either increasing drop\_out, batch normalization or L1/L2 regularization.

We will increase the number of images per video to 15 as we have ample training data.

**Experiment 2:**

Changes:

1. For each video 15 frames are used for training and validation. Resulting increase in the number of total parameters to 62837. We increase the number of inputs per video to capture more data.

**Results**:

***final train accuracy***: 91.60; ***final train loss***: .23

***final validation accuracy*** : 82.83;  ***final validation loss***: .521

**Analysis**

We see that validation loss has reduced from the previous value along with increase in accuracy. But still the model is overfitting. One reason might be as we increased the number of frames per video, number of resulting parameters also increased making the model inherently more complex.

**Experiment 3**:

Changes:

1. Introduce a additional dropout layer between the two convolutional sets with dropout = 0.25
2. Making min\_lr=0.0001

**Results**

***final train accuracy***: 37%; ***final train loss***: 1.4413

***final validation accuracy***: 37.37%; ***final validation loss***: 1.4559

**Analysis**

We see that our overfitting problem is completely gone but now the model is underfitting.

**Experiment 4**

changes:

1. We change the location of our second drop-out layer from between the 2nd and 3rd convolution layer, to between 3rd and 4th.
2. We also introduce a new dropout layer between the 4th convolution layer and flatten layer.
3. The dropout probability is set to 0.05 for these two

**Results**

***final train accuracy***: 89.3%; ***final train loss*** : .2778

***final validation accuracy***: 77.78%; ***final validation loss***: .5884

**Analysis**

We see that the model has reverted back almost to the initial performance. Now we try changing the batch size, while reverting other things to initial configuration.

**Experiment 5**

Changes:

1. We increase batch size to 42
2. Reduce number of images to 10
3. Removing the two intermediate drop out layers
4. Making filter size of first two layers 5

***final train accuracy***: 92.57; ***final train loss***: .23

***final validation accuracy***: 74.75; ***final validation loss***: .6746

**Analysis**

We see that model is still overfitting (even more than the previous versions). So, we further increase batch size. Also we introduce a dense layer with drop-out to control the feature set further.

**Experiment 6**

changes:

1. Introduce another dense layer with dropout of 0.8 and make last dropout to 0.1 of size 100
2. Increase batch size to 46

***final train accuracy***: 91%; ***final train loss***: .2211

***final validation accuracy***: 85.86%; ***final validation loss***: .575

**Analysis**

We see that we have reached a decent validation accuracy without compromising on training accuracy. Still we carry out further experiments to explore other configuration.

**Experiment 7**

Changes

1. Introduce a batch normalization layer between the two sets of conv3d
2. Increasing batch size to 50
3. Increase number of images 15
4. Reducing the filter size for 2nd set to 2.

***final train accuracy***: 66%; ***final train loss:*** .8057

***final validation accuracy***: 72%; ***final validation loss***:.8620

**Experiment 8**

1. Replace dense layer with gru of size 64, with dropout after GRU=0.8
2. Change image size to 130x130
3. Make padding of first two layers same and next two valid

***final train accuracy***: 67.9%; ***final train loss:*** .7541

***final validation accuracy***: 59% ***final validation loss:***  1.17

**Analysis**

We see that the model is underfitting. So, we reduce the dropout of the dense layer to 0.4

**Experiment 9**

Changes

1. make dropout 0.4

***final train accuracy***:86.24%; ***final train loss:*** .0.4332

***final validation accuracy***: 75% ***final validation loss:***  0.8726

**Analysis**

We see that underfitting has been highly reduced. But the still the model is overfitting.So, we introduce a GRU layer of size 64 with dropout 0.2 inplace of the 100 nodded dense layer.

**Experiment 10**

Changes

1. make dropout 0.2 gru 80

***final train accuracy***: 84.72%

***final validation accuracy***: 71%

**Analysis**

We still see that overfitting has not gone. So we introduce a 2nd batch normalization. We also add a 200 layered dense network to see if we can make the model balanced.

**Experiment 11**

Changes

1. Introduce a 2nd batch normalization layer,
2. a dense layer of size 200 with dropout 0.1

***final train accuracy***:82.53%; ***final train loss:*** .0.4552

***final validation accuracy***: 53% ***final validation loss:***  1.33

**Analysis**

We see that the overfitting increases. Implying our newly added dense layer is increasing the overfitting. But if we look at our past experiments we see that a single dense layer has performed better than the GRU. So, we drop the gru layer.

**Experiment 12**

Changes

1. Removing the GRU layer

***final train accuracy***:99.57%; ***final train loss:*** .0.0568

***final validation accuracy***: 67% ***final validation loss:***  1.0114

So, we see that we have achieved a model at one extreme. With almost completely overfitted on the training set. Therefore we increase the dropout before the 200 nodded dense layer to 0.4.

**Experiment 13**

Changes

1. dropout 0.4

***final train accuracy***:85.24%; ***final train loss:*** .0.3923

***final validation accuracy***: 74% ***final validation loss:***  .7092

**Experiment 14**

Changes

1. Make dropout 0.2
2. add random image rotation

***final train accuracy***:87.21%; ***final train loss:*** .0.3901

***final validation accuracy***: 78% ***final validation loss:***  .6091

**Conclusion**

After performing 14 experiments we see that the model in experiment 6 is performing the best. So, we select the 6th model as the final one.