Two different architectures have been tried:

1. Multi-layer Conv3D (3 models in total)
2. Conv2d stacked with a GRU (1 model)

The CNN-RNN stack produced errors with the version of TensorFlow on NimbleBox.ai (version 2.4). However, the same model ran OK with version 2.6 of tensorflow on my laptop. This however took a long time to train and consequently a smaller number of experiments were run.

The results of various experiments are tabulated below with additional details and graphs of the results on the pages following the table.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment  Number** | **Model** | **Result** | **Decision + Explanation** |
| 1 | Conv3D | Overfitting after 5 Epochs Test accuracy reach nearly 100% whilst val accuracy peaked at just under 80% | Started with image size of 128x128, and each channel normalized by dividing by the max value of the channel |
| 2 | Conv3D | Num trainable params dropped from 7.4m to 4m without any noticeable drop in attained accuracy | To reduce model size we tried reducing the image size to 96x96, and take only every second image from the 30 frames in each video sequence |
| 3 | Conv3D | Batch size of 32 was found to give shortest training time per epoch and a fewer number of epochs to reach same level of performance for larger batch sizes | Training time was still quite long (on laptop) so tried different values for batch size [32,64,96]  Batch size of 32 was found to be good compromise between time per epoch and number of epochs needed to reach acceptable performance |
| 4 | Conv3D | Dropout of 0.15, 0.15 and 0.25 were found to reduce overfitting without adversely affect the val\_accuracy  The number of epoch needed to be increased as model was learning at a slower rate | Try reduce over fitting by introducing dropout layers in the model. 3 dropout layers were added: 1 after each cond3d+max pooling block and 1 after the hidden dense layer. Various values for drop out were tried |
| 5 | Conv3D | There seemed to a slight reduction in accuracy of the model from between 60 to 70 % val accuracy to around 58 to 60% val accuracy | Different kernel size were tried to try and improve the accuracy. Combinations of 5x5x5 and 3x3x3 convolutions were tried |
| 6 | Conv3D | Lower learning rates for the Adam optimizer allow for slightly higher val\_accuracy but require a larger number of epochs to train | research on stackoverflow suggested that the Adam optimizer gets best results with a learning rate of 0.0003. Both LR=0.0003 and 0.0002 were tried. Both had a positive effect |
| 7 | Conv3D | The best result was obtained with mormalisation by dividing by np.max() for each channel  Test Accuracy = 0.856  Val Accuracy = 0.672 | Different type of image normalization and standardization were tried. Dividing each chan by np.max() and dividing each chan by 255; also standardizing by subtracting the mean and dividing by stdev were all tried |
| 8 | Conv3D | Performance was considerably poorer. So this approach was abandoned | Research on stackoverflow [<https://stackoverflow.com/questions/57467193/why-do-my-models-shift-between-100-accuracy-and-60-accuracy/57467410#57467410>] suggested changing the batch size might help when images are standardized |
| 9 | Conv3D | Over fitting now reduced and test accuracy = 75% and val accuracy around 66% | Tried RMSprop optimizer instead of Adam  Additionally tried lower LR but did not work so reduced final dense layer to 32 and added additional drop out (was 0.25, now 0.5) |
| 10 | Conv3D | Eventually managed test accuracy of around 67% and Val accuracy of around 60%  Adam and RMSprop yield better results in fewer epochs | Identical to above but with SGD optimizer (instead of RMSprop)  Had to increase LR to 0.005 and increase the number of Epochs to 30 |
| 11 | Conv3D | Training params increased to 38,864,261  Achieved least amount of over fitting with test accuracy of 71.4%  And val accuracy of 71.2% | Tried a more complex model with 3 convolutional blocks (instead of 2) with batch normalization.  It was noted that with batch normalization the training epochs need to be doubled to 40 and the val\_loss would initially rise before eventually converging to the train\_loss in the latter epochs. |
| 12 | Small Conv3D | Val\_acc – 39%, train\_acc – 52% | Decided to try and reduce params as much as possible and see if comparable accuracy could be achieved  Very simple model with 2 con3d layers, a batch\_norm layer and a hidden dense layer and a softmax output layer |
| 13 | Small Conv3D | Val\_acc – 44%  train\_acc – 57% | Added maxpool after each layer, trying to trim params |
| 14 | Small Conv3D | Val\_acc – 28%  train\_acc – 43% | Modified callback for LR to include cooldown, and factor to allow for learning to change when it stagnates |
| 15 | Small Conv3D | Extremely long epochs 1 hr+ | maxpool changed to global averagepool to see if this changed speed of steps |
| 16 | Small Conv3D | Error in first few epochs after related shape error slowing down processing | Changed shape of the pool layer from 2,2,2 to 1,2,2 |
| 17 | Small Conv3D | Val\_acc 51%  train\_acc – 68% | BN utilities added, axis, epsilon, beta and gamma: normalize along specific feature axis, make sure not dividing by zero, beta is offset factor, and gamma is scaling factor |

|  |  |  |  |
| --- | --- | --- | --- |
| 18 | Conv3D | Val\_acc – 20%  train\_acc – 56 %  Parameters < 1m | Dropped last dense layer from 128 connections to 64 to trim parameters further  Parameters  Chart, line chart  Description automatically generated  Chart, line chart  Description automatically generated |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 19 | CNN-RNN | The best that could be achieved was a train\_accuracy of 30 to 35% and a val\_accuracy of 28% | A VGG16 base model with weight initialized to “imagenet”. A Dense layer was added and that model run as a time-distributed block of layers and fed as input to a RNN using 2 GRU layers and final dense layer with softmax activation for the output classification layer |
|  |  | The numblebox.ai platform runs with Tesorflow 2.4 which seems to have a bug in running with time-distributed layers. Hence I could only run the model on my laptop with tensorflow 2.6 which led to excessively long training times.  For this reason and the poor performance the model was abandoned in favor of the CNN Conv3D model | |
| Final Model | Conv3D | Train accuracy = 86.5%  **Val accuracy = 82%**  **Total trainable params = 4,023,301**  .h5 file provided | Taking all of the best results above we ran a model with 2 conv3d blocks, dropouts, a dense\_64 layers, dropouts and a final softmax layer.  Hyper params were:  Batch size 32  Optimizer Adam  Learning rate 0.0003  Conv Kernals all 3x3x3  Dropouts of 0.15, 0.15, 0.5  Epochs = 30  Normalization of image channels using /np.max()  Image resizing to 96x96 with center crop for the 120x160 images |

**Details of the models, hyper parameters and the accuracy and loss graphs for each experiment are listed on the following pages.**

Experiment 1:

Dataset:

Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 128x128. All images resized to 128 wide and then if required black pixel padding added top and bottom to make height of image 128

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32

Model:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 13, 126, 126, 32) 2624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 6, 63, 63, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 4, 61, 61, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 2, 30, 30, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 115200) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 64) 7372864

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 325

=================================================================

Total params: 7,431,173

Trainable params: 7,431,173

Non-trainable params: 0

Optimizer: Adam with learning rate of 0.001

Num Epochs: 10

Training time: ~260secs per epoch and 15s/step

Results: Overfitting after 5 Epochs

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

Experiment 2

Whilst the model is clearly overfitting we will first try to reduce the size to both reduce number of params and to reduce the training time. In this experiment the image size will be reduced from 128x128 to 96x96 (43% reduction). Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 96x96. All images resized to 96 wide and then if required black pixel padding added top and bottom to make height of image 96

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32

Model:

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d\_4 (Conv3D) (None, 13, 94, 94, 32) 2624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_4 (MaxPooling3 (None, 6, 47, 47, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_5 (Conv3D) (None, 4, 45, 45, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_5 (MaxPooling3 (None, 2, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_2 (Flatten) (None, 61952) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 64) 3964992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 5) 325

=================================================================

Total params: 4,023,301

Trainable params: 4,023,301

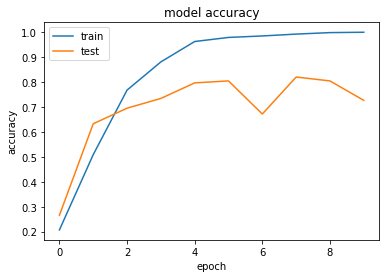
Non-trainable params: 0

Optimizer: Adam with learning rate of 0.001

Num Epochs: 10

Training time: ~165secs per epoch and 8s/step

Result: Much better training time (almost half as much time) with 3 million less params and very similar training and validation accuracy results over the 10 eopchs

Chart, line chart

Description automatically generatedShape, rectangle

Description automatically generated

Experiment 3

Whilst the model is clearly overfitting we will first try to reduce the training time. In this experiment the different Batch sizes will be tried.

Previous experiments were with batch size of 32. We will now try 64,96,128

Dataset:

Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 96x96. All images resized to 96 wide and then if required black pixel padding added top and bottom to make height of image 96

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32, 64, 96, 128

Model:

Model: "sequential\_2"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d\_4 (Conv3D) (None, 13, 94, 94, 32) 2624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_4 (MaxPooling3 (None, 6, 47, 47, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_5 (Conv3D) (None, 4, 45, 45, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_5 (MaxPooling3 (None, 2, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_2 (Flatten) (None, 61952) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_4 (Dense) (None, 64) 3964992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_5 (Dense) (None, 5) 325

=================================================================

Total params: 4,023,301

Trainable params: 4,023,301

Non-trainable params: 0

Optimizer: Adam with learning rate of 0.001

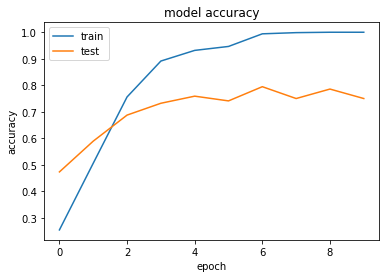
Num Epochs: 10

Result:

|  |  |
| --- | --- |
| Batch Size | Training Time |
| 16 | 170sec and 4s/step |
| 32 | 164sec and 8s/step |
| 64 | 174sec and 16s/step |
| 96 | 174sec and 25s/step |

Batch size: 32 Epoch training time is 164sec and 8s/step

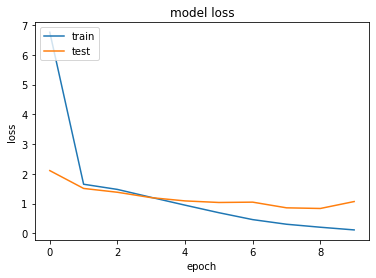
Batch Size = 32

Chart, line chart

Description automatically generated

Batch Size = 96

Chart, line chart

Description automatically generated

Experiment 4

We will now attempt to prevent the model overfitting by adding some dropout layers. Three drop out layers will be added. One after each Conv3D-maxPooling block and one after the hidden dense layer

Different values of Drop out will be tried

Dataset:

Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 96x96. All images resized to 96 wide and then if required black pixel padding added top and bottom to make height of image 96

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32

Model:

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 13, 94, 94, 32) 2624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 6, 47, 47, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 6, 47, 47, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 4, 45, 45, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 2, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 2, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 61952) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 64) 3964992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 325

=================================================================

Total params: 4,023,301

Trainable params: 4,023,301

Non-trainable params: 0

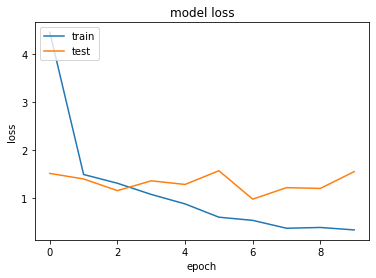
Optimizer: Adam with learning rate of 0.001

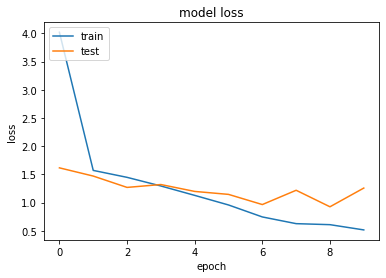
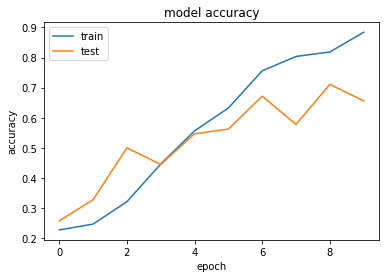
Num Epochs: 10

Results:

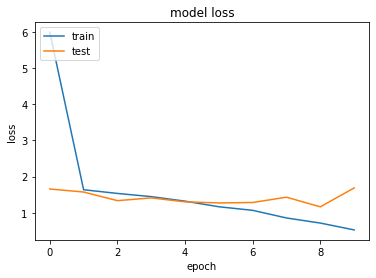
|  |  |  |  |
| --- | --- | --- | --- |
| Dropouts | Test Accuracy | Val Accuracy | Epoch where val\_loss Increases |
| 0.05;0.05;0.1 | 94.49 | 65.63 | 3 |
| 0.1;0.1;0.2 | 88.39 | 65.63 | 6 |
| 0.15;0.15;0.25 | 84.52 | 50.78 | 9 |
| 0.15;0.15;0.25  (20 epochs) | 97.32 | 60.94 | 10 |

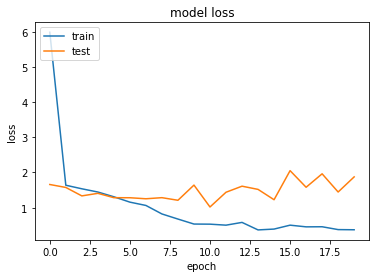
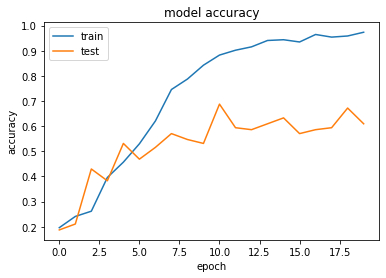
Chart, line chart

Description automatically generated



Chart, line chart

Description automatically generated



Experiment 5

We will now attempt to improve model accuracy by modifying the convolutional layers

Different kernel sizes will be tried

Dataset:

Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 96x96. All images resized to 96 wide and then if required black pixel padding added top and bottom to make height of image 96

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32

Model:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 11, 92, 92, 32) 12032

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 5, 46, 46, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 5, 46, 46, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 3, 44, 44, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 1, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 1, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 30976) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 64) 1982528

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 325

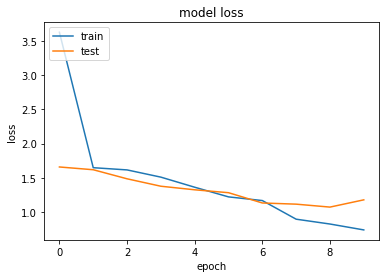
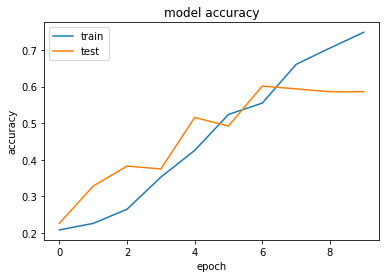
=================================================================

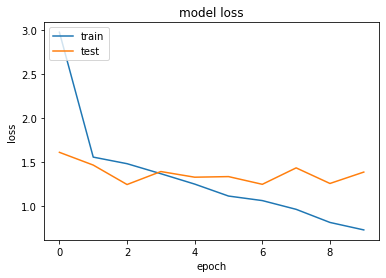
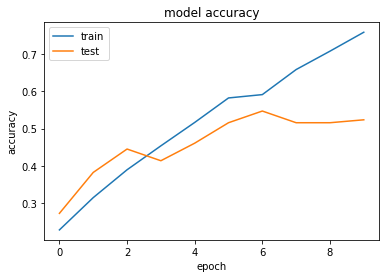
Total params: 2,050,245

Trainable params: 2,050,245

Non-trainable params: 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kernel | Test Accuracy | Val Accuracy | Overfitting |  |
| 5x5x5; 3x3x3 | 74.85 | 58.59 | Epoch 7 |  |
| 3x3x3; 5x5x5 | 75.74 | 52.34 | Epoch 6 |  |
| 5x5x5; 5x5x5 |  |  |  | (not possible) |
|  |  |  |  |  |





Experiment 6

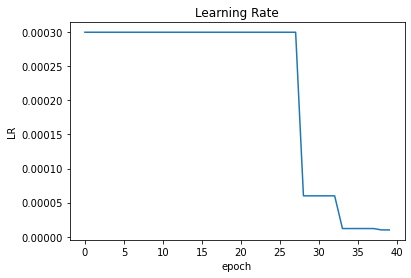
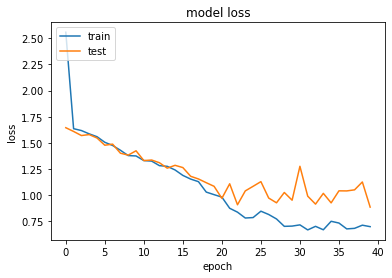
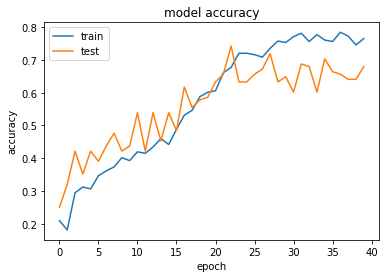
There is still evidence of over fitting so learning rate has been reduced to .0003 and the lower bound for the Reduce LR callback is now:

ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=5, min\_lr=0.00001)

Same model with 40 epochs:

Test Accuracy = 0.7648809552192688

Val Accuracy = 0.6796875

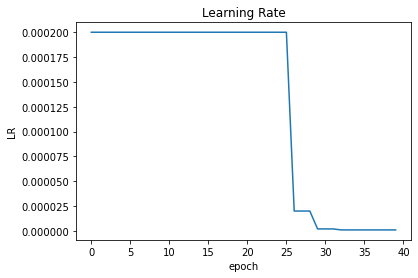
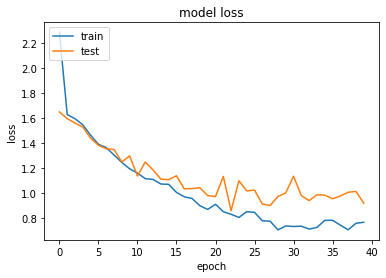
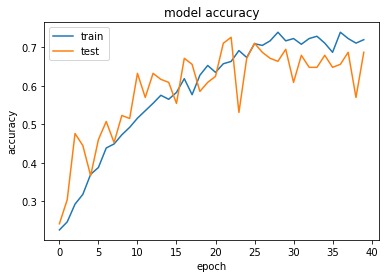


ReduceLROnPlateau(monitor='val\_loss', factor=0.1, patience=5, min\_lr=0.00001)

Same model with 40 epochs:

learning\_rate=0.0002

ReduceLROnPlateau(monitor='val\_loss', factor=0.1,patience=3, min\_lr=0.000001)



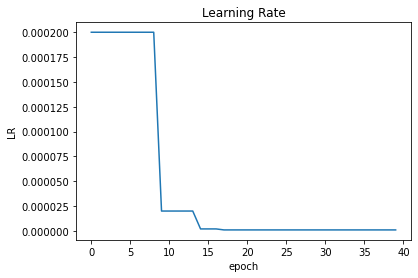
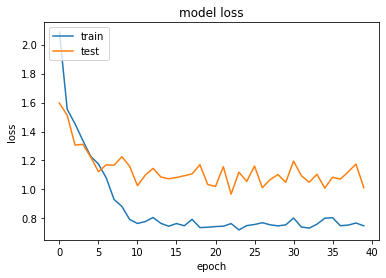
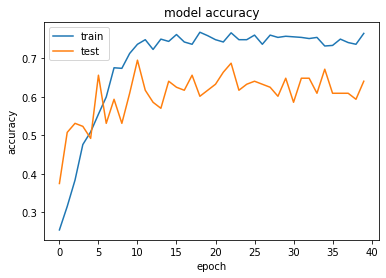
30 EPOCHS looks about max: now we’ll loosen up the model with less drop out to try and get even better accuracy

Experiment 7

Same model same input data but with slightly less drop outs (from .15,.15,.25 down to .1,.1,.2

Test Accuracy = 0.7648809552192688

Val Accuracy = 0.640625



Experiment 8

Test different ways of normailing the images

1. Baseline is each channel by /np.max
2. Image /255
3. Standardize each channel

Dataset:

Video sequences: Every 2nd frame , so 15 frames from each sequence

Image sizes: 96x96. All images resized to 128 wide and then if required black pixel padding added top and bottom to make height of image 96

Image Normalizations: 0 to 1: each channel of each image is divided by the max value for that channel

Batch size: 32

Model:

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 11, 92, 92, 32) 12032

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 5, 46, 46, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 5, 46, 46, 32) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 3, 44, 44, 64) 55360

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 1, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 1, 22, 22, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 30976) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 64) 1982528

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 325

=================================================================

Total params: 2,050,245

Trainable params: 2,050,245

Non-trainable params: 0

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.0002

Each channel divided by np.max()

Test Accuracy = 0.855654776096344

Val Accuracy = 0.671875

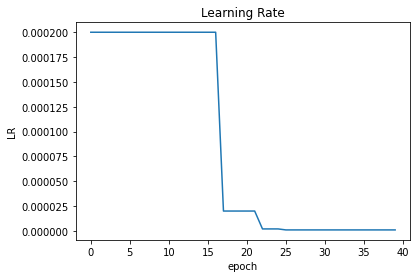
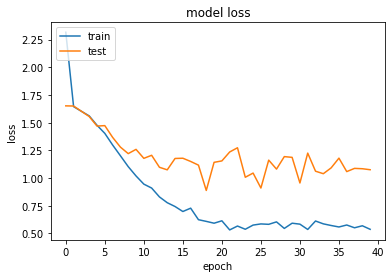
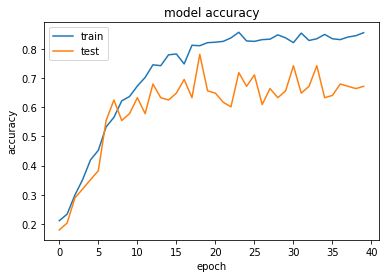


Image /255

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

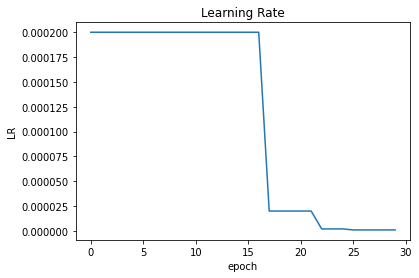
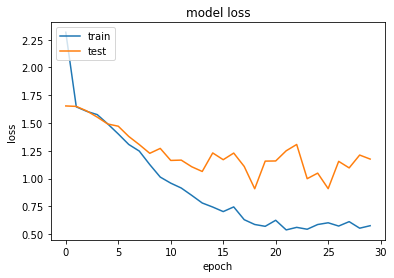
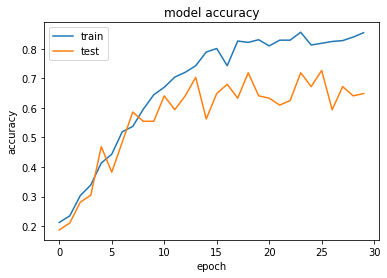
# epochs = 30

# Batch Size = 32

# Initial LR = 0.0002

Test Accuracy = 0.8541666865348816

Val Accuracy = 0.6484375



HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 30

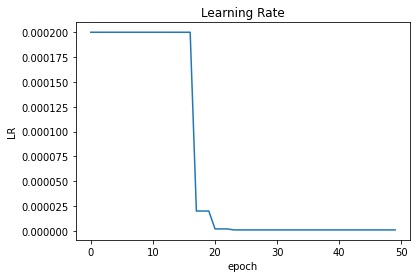
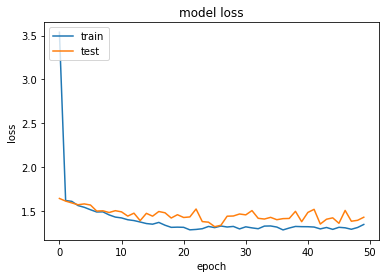
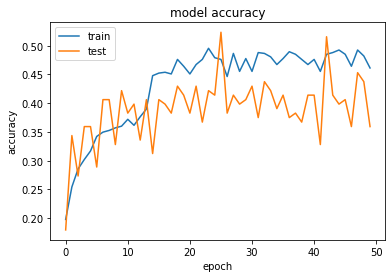
# Batch Size = 32

# Initial LR = 0.001

Each channel is /255 and then each channel is standardized (X-mean(X)) / stdv (X)

Test Accuracy = 0.4613095223903656

Val Accuracy = 0.359375



HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 50

# Batch Size = 32

# Initial LR = 0.0002

Conclusion: looks to get stuck in local minima. Tried increasing batch size per recommendations from stackoverflow, but this made it worse! [ <https://stackoverflow.com/questions/57467193/why-do-my-models-shift-between-100-accuracy-and-60-accuracy/57467410#57467410> ]

Experiment 9

So instead looked to change the Optimizer from adam to RMSprop

Test Accuracy = 0.9702380895614624

Val Accuracy = 0.546875

Chart, line chart

Description automatically generated Chart, line chart

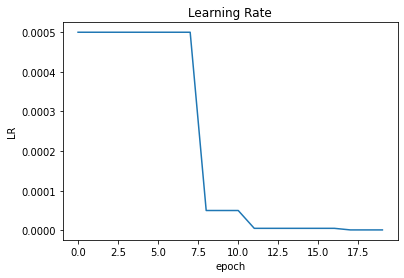
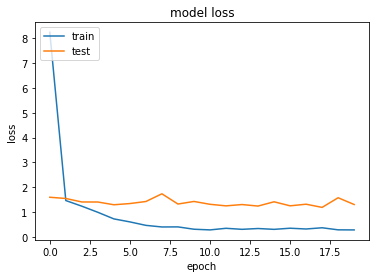
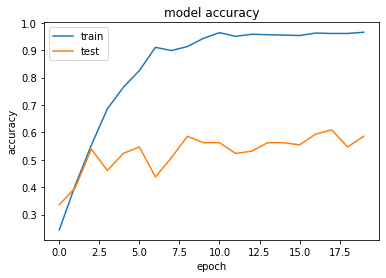
Description automatically generatedChart, line chart

Description automatically generated

There is clear evidence of overfitting, so can increase dropout (regularization) and reduce learning rate from default 0.001

Test Accuracy = 0.9657738208770752

Val Accuracy = 0.5859375



HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 20

# Batch Size = 32

# Initial LR = 0.0005

Interesting with lower learning rate the val\_score is worse, so will try changing the model instead:

Smaller dense layer (64 -> 32):

Experiment: 9 continued

model changed with dense layer of 32 (instead of 64)

Test Accuracy = 0.7827380895614624

Val Accuracy = 0.609375

Avg Test Accuracy = 0.7275132338205973

Avg Val Accuracy = 0.5746527777777778

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generated

Shape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 20

# Batch Size = 32

# Initial LR = 0.0005

With additional drop out (0.5 instead of 0.25) after final dense layer

Test Accuracy = 0.7514880895614624

Val Accuracy = 0.640625

Avg Test Accuracy = 0.7276785771052042

Avg Val Accuracy = 0.6605902777777778

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 20

# Batch Size = 32

# Initial LR = 0.0005

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop object at 0x7fbe300b2280>

Experiment 10

Identical to above but with SGD optimizer (instead of RMSprop)

Test Accuracy = 0.3348214328289032

Val Accuracy = 0.4609375

Avg Test Accuracy = 0.3151455024878184

Avg Val Accuracy = 0.3185763888888889

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, rectangle

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 20

# Batch Size = 32

# Initial LR = 0.0005

# Optimizer : <tensorflow.python.keras.optimizer\_v2.gradient\_descent.SGD object at 0x7f2b3790f130>

Experiment 10 continued

Test Accuracy = 0.4360119104385376

Val Accuracy = 0.5078125

Avg Test Accuracy = 0.39417989055315655

Avg Val Accuracy = 0.4704861111111111

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.001

# Optimizer : <tensorflow.python.keras.optimizer\_v2.gradient\_descent.SGD object at 0x7fd915592400>

Experiment 10 continued

Test Accuracy = 0.6473214030265808

Val Accuracy = 0.59375

Avg Test Accuracy = 0.6301256616910299

Avg Val Accuracy = 0.5920138888888888

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.005

# Optimizer : <tensorflow.python.keras.optimizer\_v2.gradient\_descent.SGD object at 0x7fdfb41aca90>

Experiment 11 :

More complex model (3 conv3d blocks with batch normalization

Initially trained with normalized images (each chan / 255)

And SGD optimizer LR=0.005

Model: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 15, 96, 96, 64) 5248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization (BatchNo (None, 15, 96, 96, 64) 256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation (Activation) (None, 15, 96, 96, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 7, 48, 96, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 7, 48, 96, 128) 221312

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_1 (Batch (None, 7, 48, 96, 128) 512

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_1 (Activation) (None, 7, 48, 96, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 3, 24, 48, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_2 (Conv3D) (None, 3, 24, 48, 256) 884992

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_2 (Batch (None, 3, 24, 48, 256) 1024

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation) (None, 3, 24, 48, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_2 (MaxPooling3 (None, 1, 12, 24, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 73728) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 73728) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 512) 37749248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 512) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 2565

=================================================================

Total params: 38,865,157

Trainable params: 38,864,261

Test Accuracy = 0.6726190447807312

Val Accuracy = 0.71875

Avg Test Accuracy = 0.6598875522613525

Avg Val Accuracy = 0.6944444444444444

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape

Description automatically generated with medium confidence

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.005

Experiment 11 continued

Reduced initial learning rate, same model.

Test Accuracy = 0.644345223903656

Val Accuracy = 0.6953125

Avg Test Accuracy = 0.6569113797611661

Avg Val Accuracy = 0.6831597222222222

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.0005

# Optimizer : <tensorflow.python.keras.optimizer\_v2.gradient\_descent.SGD

Exp 11 cont.

Same model mow with 120x160 images cropped to 120x120 before being resized to 96x96

Test Accuracy = 0.7142857313156128

Val Accuracy = 0.7109375

Avg Test Accuracy = 0.7141203681627909

Avg Val Accuracy = 0.703125

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 0.0005

# Optimizer : <tensorflow.python.keras.optimizer\_v2.gradient\_descent.SGD

Exp 11 cont.

Exactly same as previous but image first normalized then centred by subtracting mean. Also used RMSprop optimizer

Test Accuracy = 0.8928571343421936

Val Accuracy = 0.7578125

Avg Test Accuracy = 0.8763227462768555

Avg Val Accuracy = 0.7109375

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 5e-05

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop object at 0x7fe9388e7a30>

Exp11 cont

Identical to last but without any batch normalization layers

Test Accuracy = 0.9598214030265808

Val Accuracy = 0.7421875

Avg Test Accuracy = 0.9550264610184563

Avg Val Accuracy = 0.7430555555555556

A picture containing graphical user interface

Description automatically generatedA picture containing application

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 40

# Batch Size = 32

# Initial LR = 5e-05

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop

Test Accuracy = 0.9211309552192688

Val Accuracy = 0.6640625

Avg Test Accuracy = 0.9255952371491326

Avg Val Accuracy = 0.6605902777777778

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedA picture containing shape

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 20

# Batch Size = 32

# Initial LR = 5e-05

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop

Exp 12:

3 conv3d blocks with 5x5x5 kernels

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv3d (Conv3D) (None, 15, 96, 96, 64) 24064

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation (Activation) (None, 15, 96, 96, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d (MaxPooling3D) (None, 7, 48, 96, 64) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_1 (Conv3D) (None, 7, 48, 96, 128) 1024128

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_1 (Activation) (None, 7, 48, 96, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_1 (MaxPooling3 (None, 3, 24, 48, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 3, 24, 48, 128) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_2 (Conv3D) (None, 3, 24, 48, 256) 4096256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation) (None, 3, 24, 48, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_2 (MaxPooling3 (None, 1, 12, 24, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout) (None, 1, 12, 24, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten) (None, 73728) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_2 (Dropout) (None, 73728) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense) (None, 256) 18874624

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_3 (Dropout) (None, 256) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 5) 1285

=================================================================

Total params: 24,020,357

Trainable params: 24,020,357

Test Accuracy = 0.944940447807312

Val Accuracy = 0.734375

Avg Test Accuracy = 0.8548280464278327

Avg Val Accuracy = 0.7430555555555556

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedA picture containing shape

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 15

# Batch Size = 32

# Initial LR = 0.0001

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop

Experiment 11 continued

Same as above with call backs to save the model for best val\_loss

Test Accuracy = 0.9479166865348816

Val Accuracy = 0.828125

Avg Test Accuracy = 0.8505291011598375

Avg Val Accuracy = 0.7352430555555556

Chart, line chart

Description automatically generatedChart, line chart

Description automatically generatedShape, square

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 15

# Batch Size = 32

# Initial LR = 0.0001

# Optimizer : <tensorflow.python.keras.optimizer\_v2.rmsprop.RMSprop

CNN-RNN stack

Base model is VGG16 with imagenet weights + dense(64)

Layer (type) Output Shape Param #

=================================================================

time\_distributed (TimeDistri (None, 15, 64) 15009664

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru (GRU) (None, 15, 32) 9408

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

gru\_1 (GRU) (None, 16) 2400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout) (None, 16) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense) (None, 8) 136

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense) (None, 5) 45

=================================================================

Total params: 15,021,653

Trainable params: 306,965

Non-trainable params: 14,714,688

Test Accuracy = 0.244047611951828

Val Accuracy = 0.375

Avg Test Accuracy = 0.2683531741301219

Avg Val Accuracy = 0.3246527777777778

A picture containing company name

Description automatically generatedA picture containing company name

Description automatically generatedA picture containing shape

Description automatically generated

HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 50

# Batch Size = 32

# Initial LR = 0.001

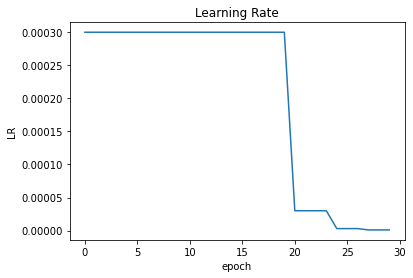
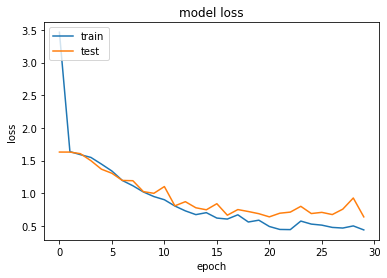
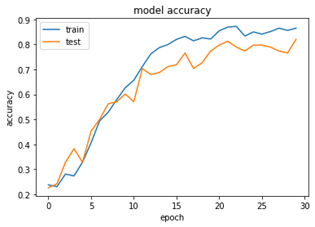
Final Model

Test Accuracy = 0.8645833134651184

Val Accuracy = 0.8203125

Avg Test Accuracy = 0.8544973532358805

Avg Val Accuracy = 0.7881944444444444



HYPERPARAMS

===========

# training sequences = 663

# validation sequences = 100

# epochs = 30

# Batch Size = 32

# Initial LR = 0.0003

# Optimizer : <keras.optimizer\_v2.adam.Adam