Since the associated work is with partly image processing, we have decided to use transfer learning rather than building up a new conv net from scratch

1. Augmentation of images:
    1. Flipping: Flipping the images is not chosen, Since the image classification is about 5 gestures
        1. 2 of the gestures are left-right sensitive, flip horizontally does not make sense
        2. 2 of the gestures are top down sensitive, flip vertically does not make sense
    2. Affine transformations:
        1. Real world images are to the scale and would not be transformed and hence not considered
    3. Erosion and dilation: Since the images have 3 different channels, and they are not Black and white, mathematically, erosion and dilation would not make sense about 3 different channels as it might introduce unknown colours into the images.
2. Data generation: To work with the images we have taken the size to be 120x120, we are experimenting with different image sizes as long as we are not running into Out of memory exception

We have used three pre trained networks before fixing one.
Trial 1: VGG16 which gave satisfactory results. But during the trials, the validation accuracy was found to be a little bit erratic hence we also decided to try with other pretrained models
Trial 1: ResNet V2, which took much higher training time hence we evaded this model very fast
Trial 1: MobileNet, since being a light weight pretrained model, did not take too much training time and was on par with VGG16. Yet similar erratic nature of validation accuracy was seen as that of VGG16 hence decided to finally work with VGG 16

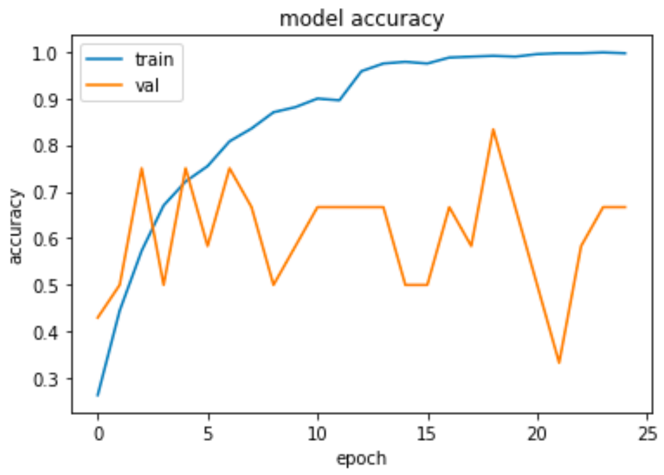
As part of trying to smoothen the Validation loss/accuracy curves of the training, we have worked with multiple sets of learning rate and figured that higher learning rates are one of the reason, hence we reset the learning rates from 0.01 to 0.001.

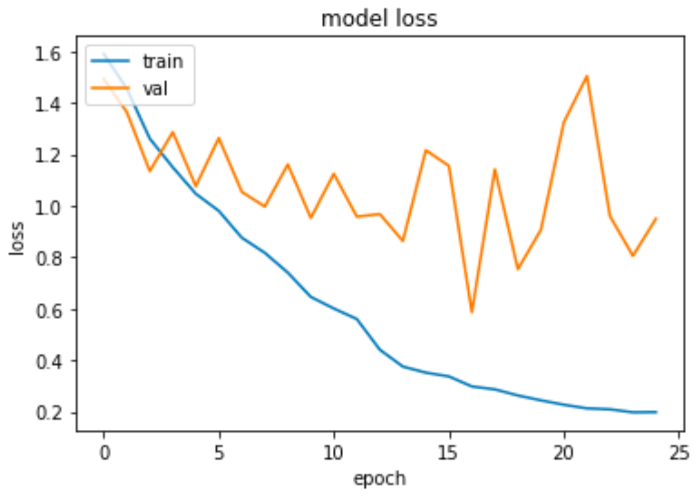
for optimizers Adam and RMSProp gave better stability for validation curves

Before fixing the model, we had to decide which possible RNN model would be a better fit.
We have examined, a regularized RNN units under multiple RNN network types and obtained the following results:

Baselining of accuracy of whole exercise is at 20% as there are 5 categories

1. Simple RNN:

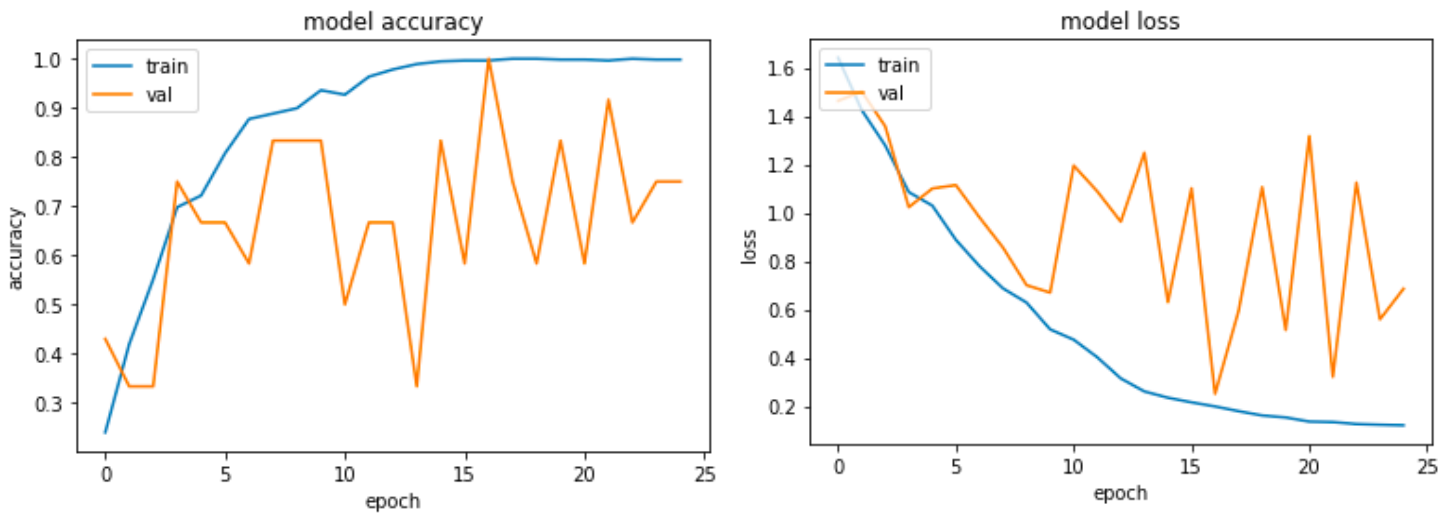




Clear case of overfitting, yet, not 100% accuracy shown with Simple RNN.
Best training loss, training accuracy, validation loss, validation accuracy :
0.29761-0.98718-0.58579-0.66667 at epoch 17

```
14/14 [==============================] - 50s 4s/step -
loss: 0.2976 - categorical_accuracy: 0.9872 - val_loss:
0.5858 - val_categorical_accuracy: 0.6667

Epoch 00017: val_loss improved from 0.86273 to 0.58579,
saving model to
VGG_Experimental_RNN_120_48_10_16_48_2020-08-1712_33_08.3
27377/model-00017-0.29761-0.98718-0.58579-0.66667.h5
```

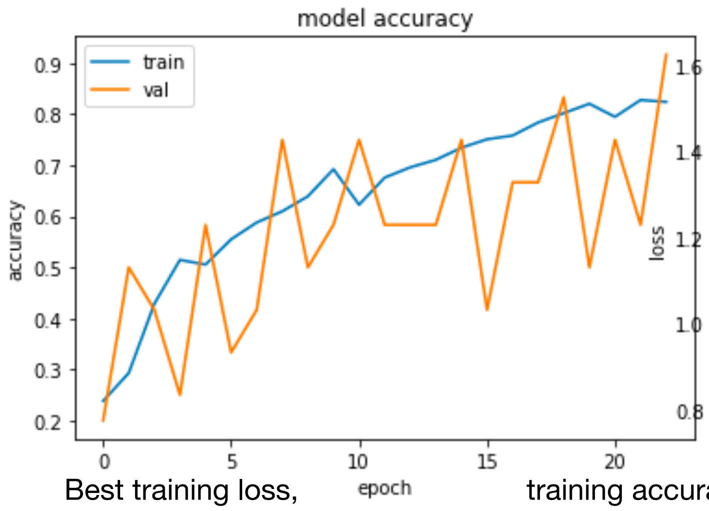
2. GRU:

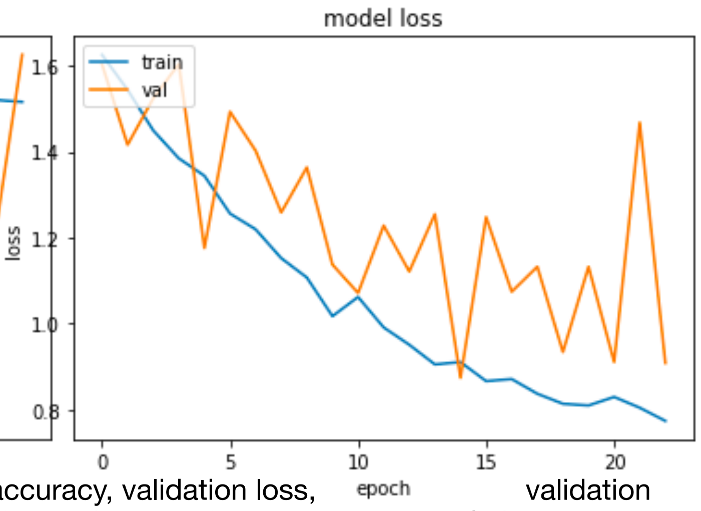


Best training loss,   training accuracy, validation loss, validation accuracy :
0.19824-0.99634-0.25035-1.00000 at epoch 17

```
Epoch 17/60

14/14 [==============================] - ETA: 0s - loss: 0.1982 - categorical_accuracy: 0.9963

Epoch 00017: val_loss improved from 0.62953 to 0.25035, saving model to

VGG_Experimental_GRU_120_48_10_16_48_2020-08-2312_07_39.713802/

model-00017-0.19824-0.99634-0.25035-1.00000.h5
```

3. LSTM:

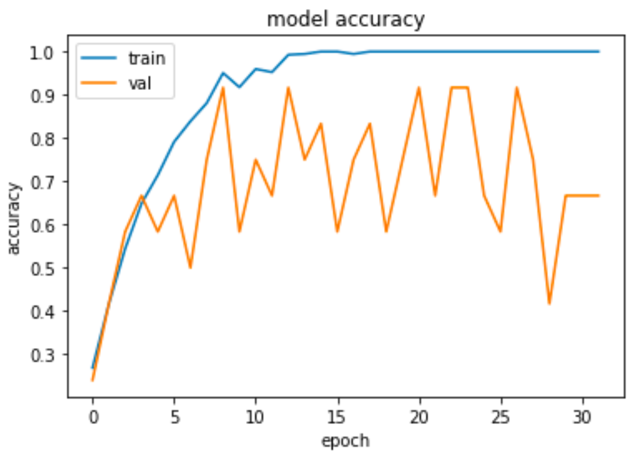


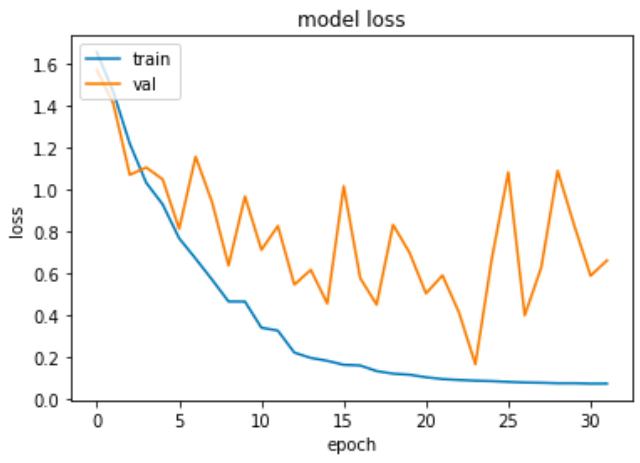


Best training loss, training accuracy, validation loss, validation accuracy : 0.91005-0.73443-0.87351-0.75000 at epoch 15

```
Epoch 00015: val_loss improved from 1.07076 to 0.87351, saving model to
VGG_Experimental_LSTM_120_48_10_16_48_2020-08-2314_08_32.278230/
model-00015-0.91005-0.73443-0.87351-0.75000.h5
14/14 [==============================] - 249s 18s/step - loss: 0.9101 - categorical_accuracy: 0.7344
- val_loss: 0.8735 - val_categorical_accuracy: 0.7500 - lr: 1.6000e-04
```

4. GRU - Bidirectional:



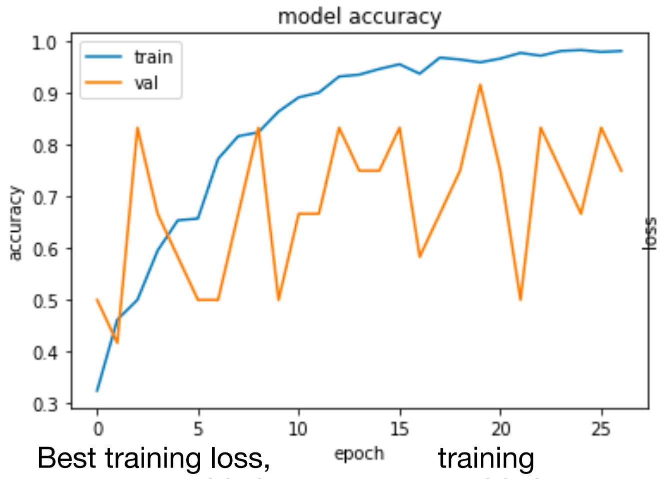


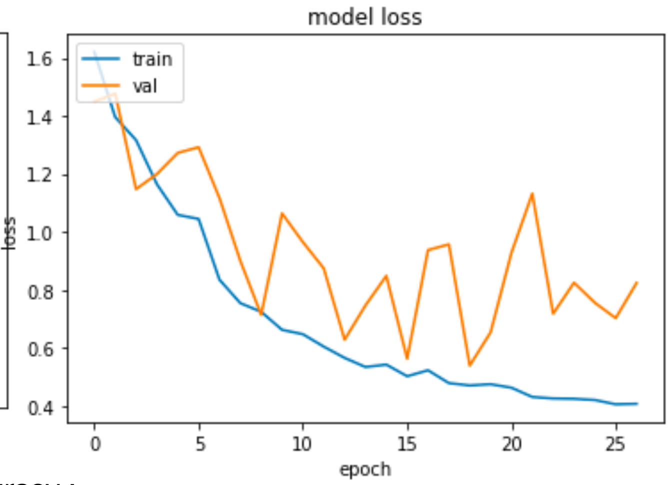
Best training loss, training accuracy, validation loss, validation accuracy :
0.08576-1.00000-0.16414-0.91667 at epoch 24

```
Epoch 24/60
14/14 [==============================] - 51s 4s/step - loss: 0.0858 - categorical_accuracy: 1.0000 -
val_loss: 0.1641 - val_categorical_accuracy: 0.9167
```

```
Epoch 00024: val_loss improved from 0.41218 to 0.16414, saving model to
VGG_Experimental_GRUbi_120_48_10_16_48_2020-08-1712_05_41.280210/
model-00024-0.08576-1.00000-0.16414-0.91667.h5
```

5. LSTM - Bidirectional:





Best training loss, training accuracy, validation loss, validation accuracy :
0.50193-0.95604-0.56254-0.83333 at epoch 16

```
14/14 [==============================] - 52s 4s/step - loss: 0.5019 - categorical_accuracy: 0.9560 -
val_loss: 0.5625 - val_categorical_accuracy: 0.8333

Epoch 00016: val_loss improved from 0.62801 to 0.56254, saving model to
VGG_Experimental_LSTMbi_120_48_10_16_48_2020-08-1712_55_32.703884/
model-00016-0.50193-0.95604-0.56254-0.83333.h5
```

It is observed that LSTM/GRU bidirectional is stabler.

Compared to GRU/LSTM, Simple RNN is thoroughly overfitting, which means we can clearly not proceed with SimpleRNN

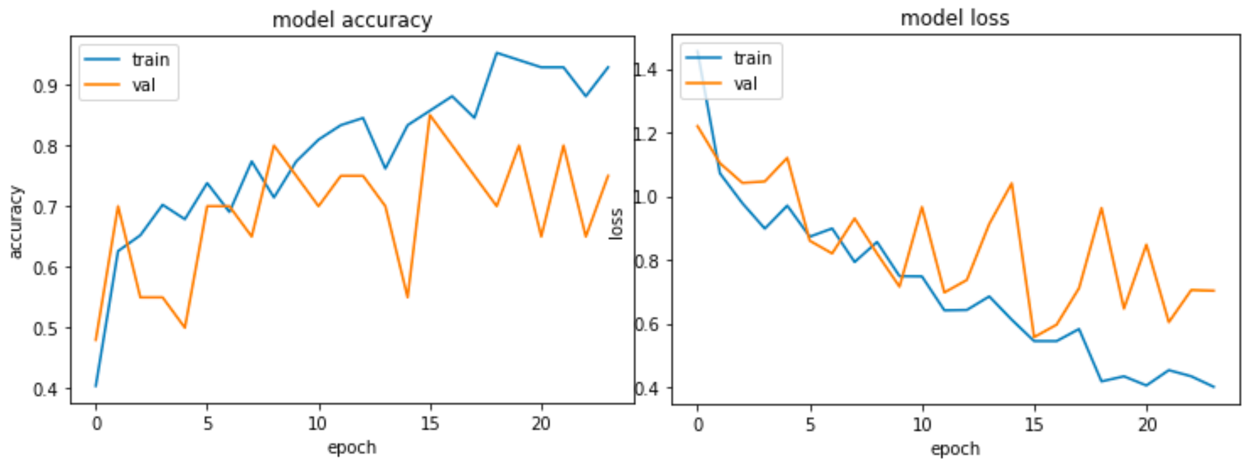
GRU and LSTM are also predictive enough to give high accuracies of near 100% on validation set.

Since the hunt is for model with least, parameters, GRU and LSTM are the better choice for task

One pearl found with GRU :
Best training loss, training accuracy, validation loss, validation accuracy :
0.19824-0.99634-0.25035-1.00000 at epoch 17

Experimenting on LSTM and GRU with different batch sizes and units:

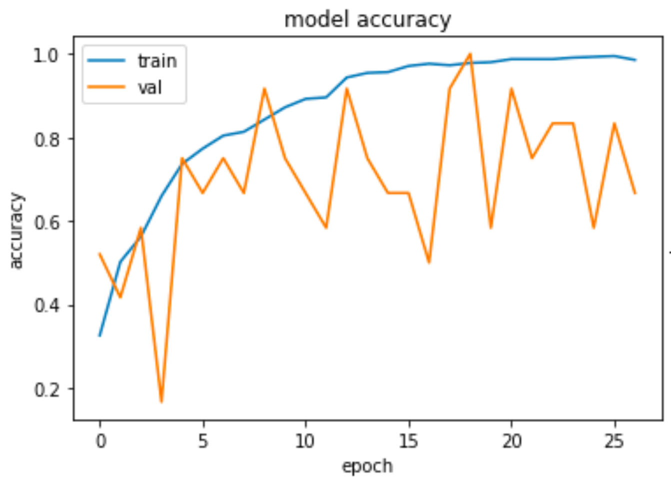
1. GRU 8 units, batch size 24 clips:

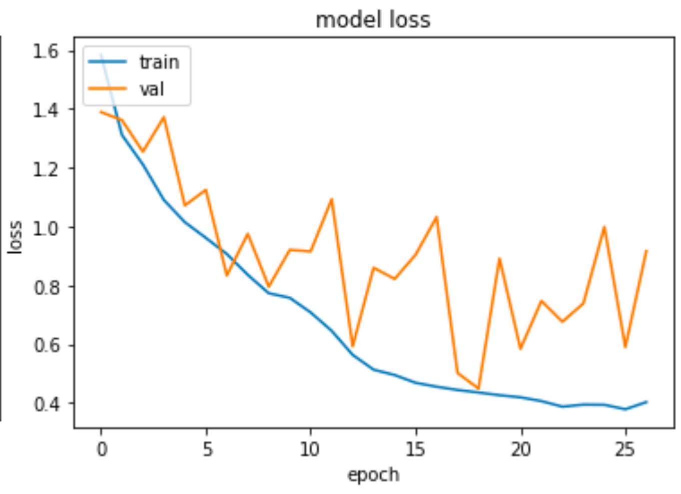


```
28/28 [==============================] - 9s 336ms/step - loss: 0.5443 - categorical_accuracy: 0.8571
- val_loss: 0.5567 - val_categorical_accuracy: 0.8500

Epoch 00016: val_loss improved from 0.69711 to 0.55667, saving model to
VGG_Experimental_GRU_120_24_8_16_24_directdense/model-00016-0.54429-0.85714-0.55667-0.85000.h5
```

Stable, consistent variation of validation loss seen

2. GRU 8 units, batch size 48 clips:



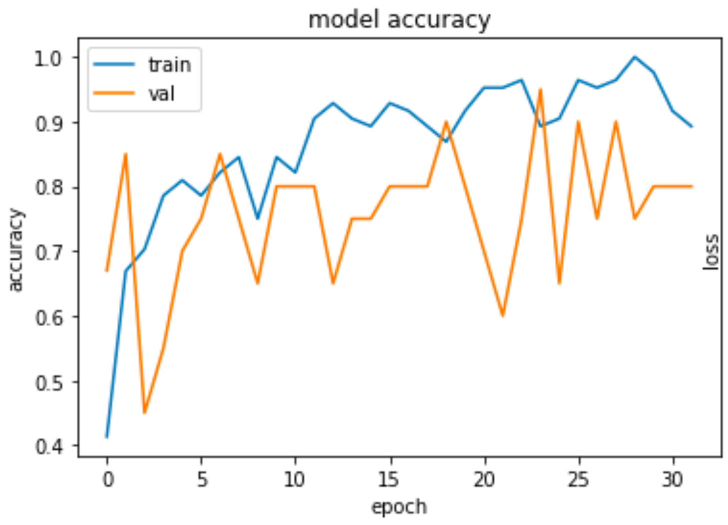


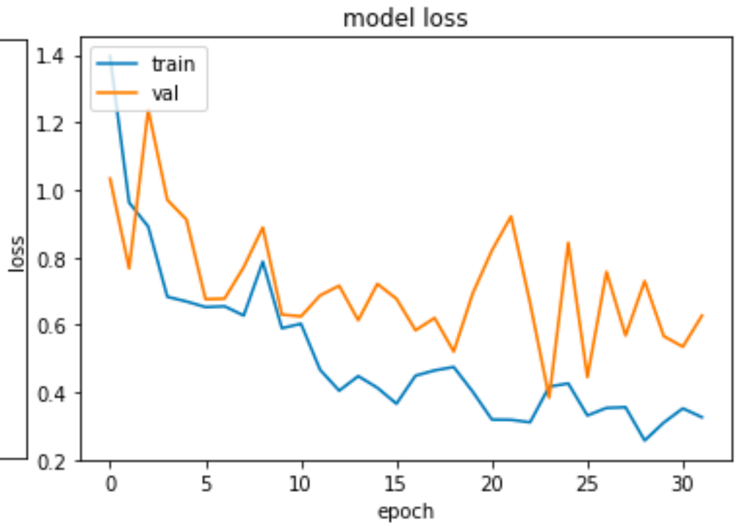
```
14/14 [==============================] - 51s 4s/step - loss: 0.4358 -
categorical_accuracy: 0.9780 - val_loss: 0.4492 - val_categorical_accuracy: 1.0000

Epoch 00019: val_loss improved from 0.50231 to 0.44918, saving model to
VGG_Experimental_GRU_120_48_8_16_48_directdense/
model-00019-0.43583-0.97802-0.44918-1.00000.h5
```

Stable, consistent variation of validation loss seen

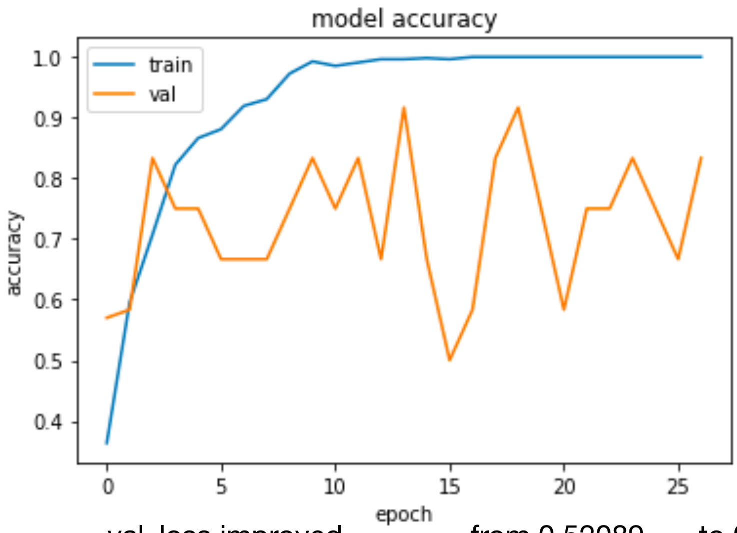
This graph is the proof that 8 unit GRU also has enough predictive power for the task, with model size as puny as 60MB

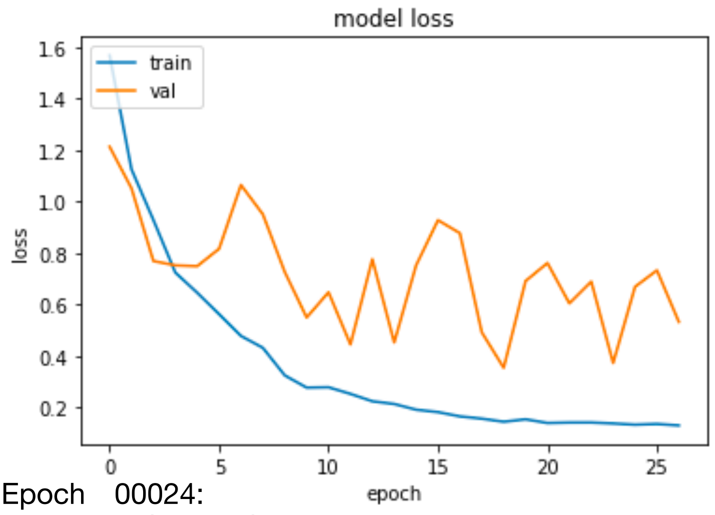
3. GRU 15 units, batch size 24 clips:





```
Epoch 24/60
28/28 [==============================] - 9s 331ms/step - loss: 0.4170 -
categorical_accuracy: 0.8929 - val_loss: 0.3839 - val_categorical_accuracy: 0.9500
```



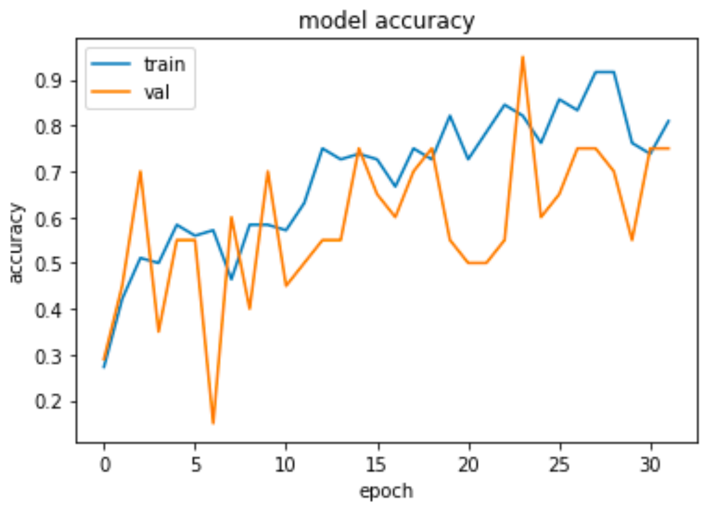


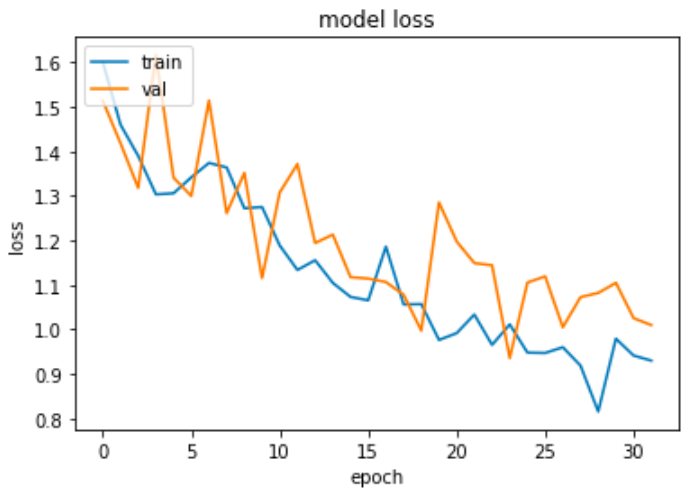
```
Epoch    00024:
val_loss improved          from 0.52089       to 0.38386, saving model to
                    VGG_Experimental_GRU_120_24_15_16_24_directdense/
model-00024-0.41700-0.89286-0.38386-0.95000.h5
```

Here, model has generalized even better for validation over training set, due to regularization

4. GRU 15 units, batch size 48 clips:

```
Epoch 19/60
14/14 [==============================] - 52s 4s/step - loss: 0.1443 -
categorical_accuracy: 1.0000 - val_loss: 0.3540 - val_categorical_accuracy: 0.9167

Epoch 00019:                    val_loss improved from 0.44463 to 0.35397,        saving model to
```





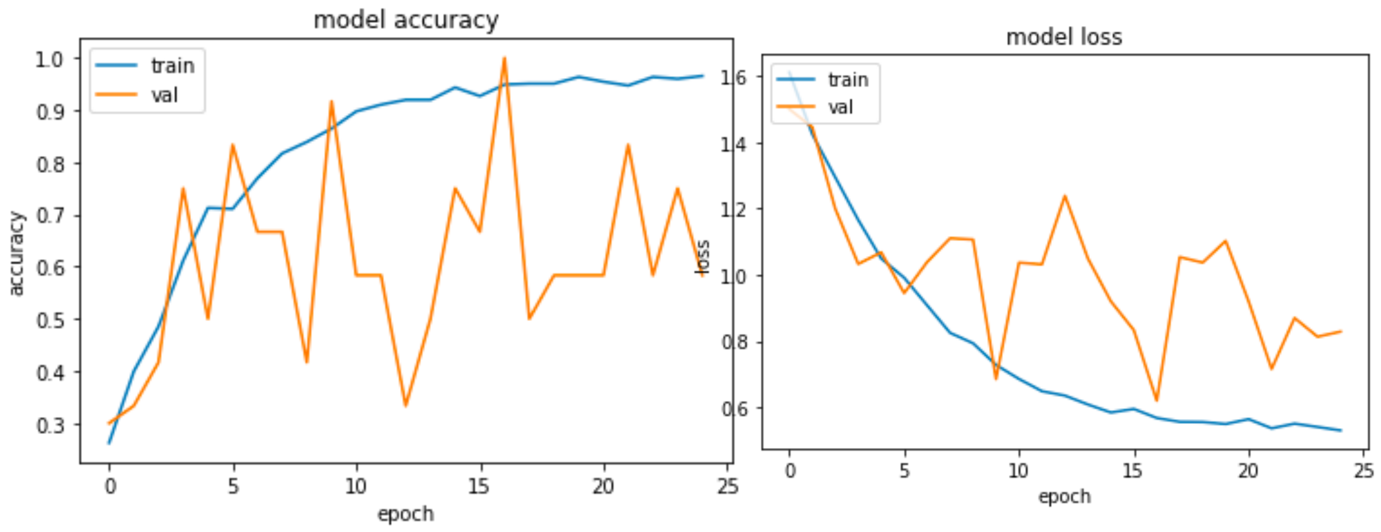
```
                    VGG_Experimental_GRU_120_48_15_16_48_directdense/
model-00019-0.14431-1.00000-0.35397-0.91667.h5
```

5. LSTM 8 units, batch size 24 clips:

```
28/28 [==============================] - 10s 355ms/step - loss: 1.0112 -
categorical_accuracy: 0.8214 - val_loss: 0.9357 - val_categorical_accuracy: 0.9500

Epoch 00024: val_loss improved from 0.99726 to 0.93573, saving model to
VGG_Experimental_LSTM_120_24_8_16_24_directdense/
model-00024-1.01124-0.82143-0.93573-0.95000.h5
```

Not impressive, can be trained for many more epochs, underfitting seen, can improve network predictive power

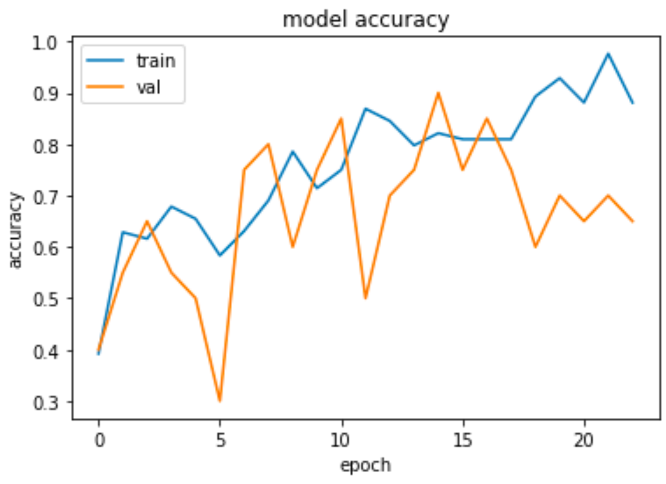
6. LSTM 8 units, batch size 48 clips:

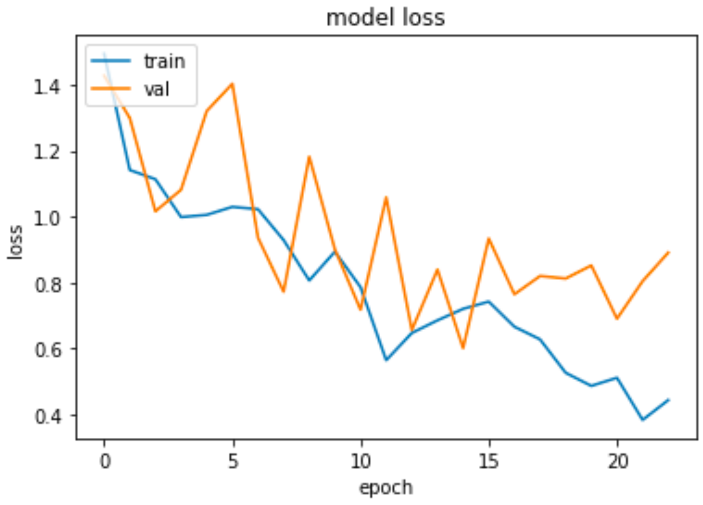


```
14/14 [==============================] - 50s 4s/step - loss: 0.5677 -
categorical_accuracy: 0.9487 - val_loss: 0.6199 - val_categorical_accuracy: 1.0000

Epoch 00017: val_loss improved from 0.68499 to 0.61987, saving model to
VGG_Experimental_LSTM_120_48_8_16_48_directdense/
model-00017-0.56773-0.94872-0.61987-1.00000.h5
```

Model training looks erratic for validation curves, yet the model generalized very well

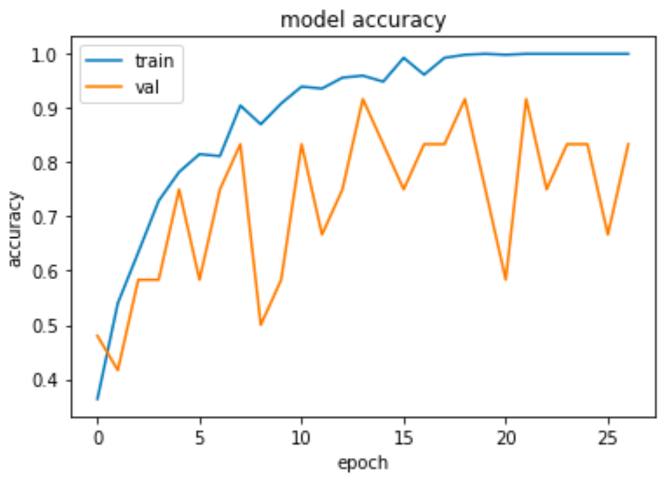
7. LSTM 15 units, batch size 24 clips:

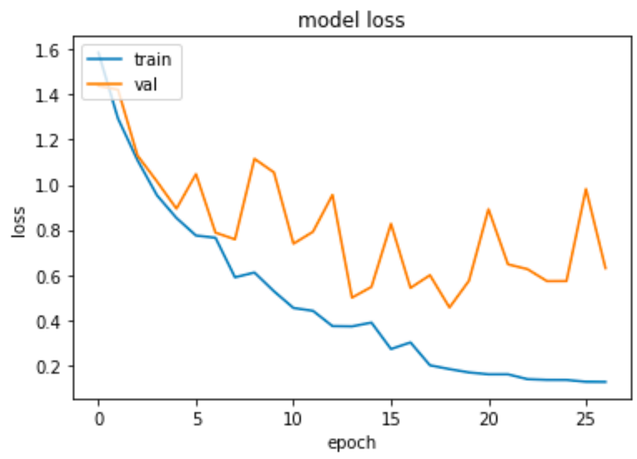




```
Epoch 00013: val_loss improved from 0.71760 to 0.65597, saving model to
VGG_Experimental_LSTM_120_24_15_16_24_directdense/
model-00013-0.64712-0.84524-0.65597-0.70000.h5
Epoch 14/60
28/28 [==============================] - 10s 352ms/step - loss: 0.6855 -
categorical_accuracy: 0.7976 - val_loss: 0.8393 - val_categorical_accuracy: 0.7500

Epoch 00014: val_loss did not improve from 0.65597
Epoch 15/60
28/28 [==============================] - 10s 340ms/step - loss: 0.7203 -
categorical_accuracy: 0.8214 - val_loss: 0.6012 - val_categorical_accuracy: 0.9000

Epoch 00015: val_loss improved from 0.65597 to 0.60119, saving model to
VGG_Experimental_LSTM_120_24_15_16_24_directdense/
model-00015-0.72029-0.82143-0.60119-0.90000.h5
```

8. LSTM 15 units, batch size 48 clips:



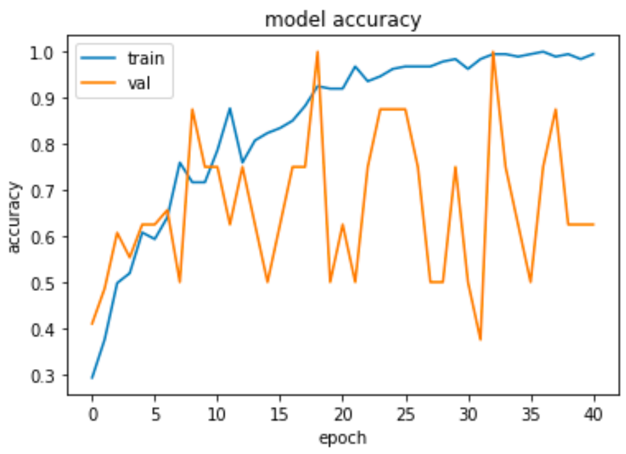


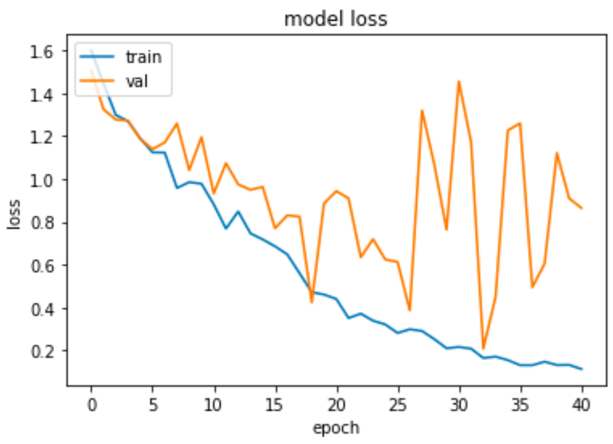
```
Epoch 00014: val_loss improved from 0.73870 to 0.49936, saving model to
VGG_Experimental_LSTM_120_48_15_16_48_directdense/
model-00014-0.37183-0.95971-0.49936-0.91667.h5

Epoch 00019: val_loss improved from 0.49936 to 0.45560, saving model to
VGG_Experimental_LSTM_120_48_15_16_48_directdense/
model-00019-0.18237-0.99817-0.45560-0.91667.h5
```

It looks like the training started to overfit between epoch 14 and 19.

We have one very good model with GRU Bidirectional : with 8 images per clip
GRU bidirectional : 6 GRU units batch size 64 Trainable params: 166,325

```
Epoch 33/60
11/11 [==============================] - 9s 844ms/step - loss: 0.1641 -
categorical_accuracy: 0.9947 - val_loss: 0.2080 - val_categorical_accuracy: 1.0000

Epoch 00019: val_loss improved from 0.76985 to 0.42363, saving model to
VGG16_MultiGRU_Bidirectional120_64_64_2020-08-1613_30_20.621711/
model-00019-0.47154-0.92513-0.42363-1.00000.h5


Epoch 00033: val_loss improved from 0.38682 to 0.20797, saving model to
VGG16_MultiGRU_Bidirectional120_64_64_2020-08-1613_30_20.621711/
model-00033-0.16406-0.99465-0.20797-1.00000.h5
```





Since the validation run is so erratic, not submitting the model.

Working with reducing the inconsistency of validation accuracy:
We have already worked with multiple batch sizes and RNN types.

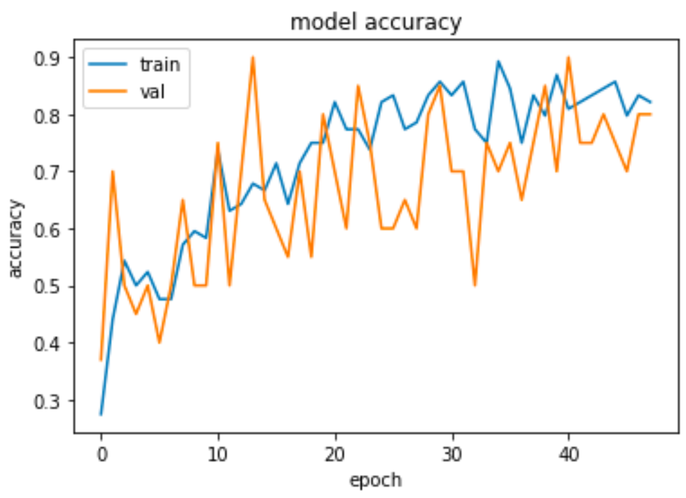
Trial 1: Setting up decay 0.01 and momentum 0.7 with SGD optimiser - Taking too long for prediction
Trial 2: Setting up decay 0.001 and momentum 0.7 with SGD optimiser - Still Taking too long for prediction
Trial 3: Setting up decay 0.0005 and momentum 0.9 with SGD optimiser - Not taking too long,validation curves running alongside training curves for both validation loss and accuracy, much better than previous optimiser with no momentum.

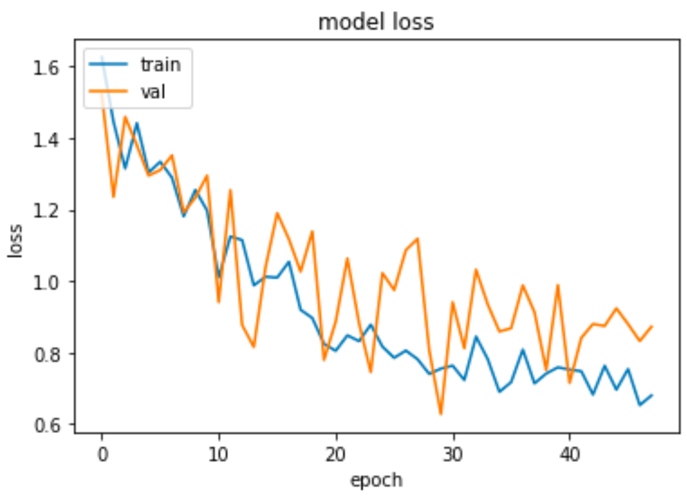
Selecting trials with GRU and Bidirectional GRU

Early stop with with no min cap on learning rate with patience of 15 epochs

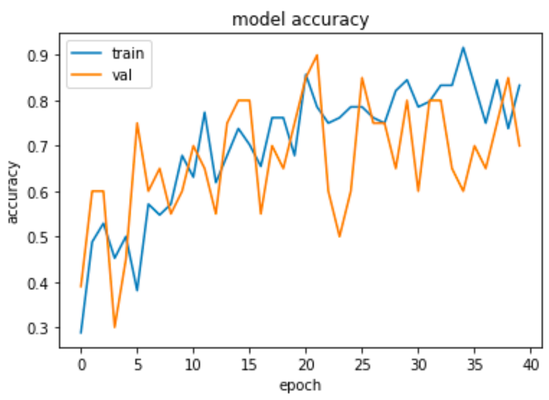
1. GRU units 15 , batch size 24:

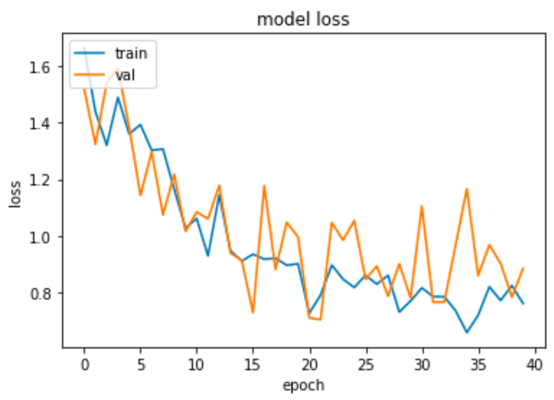
```
/28 [==============================] - 10s 340ms/step - loss: 0.7554 - categorical_accuracy: 0.8571
- val_loss: 0.6280 - val_categorical_accuracy: 0.8500

Epoch 00030: val_loss improved from 0.74550 to 0.62804, saving model to

VGG_Experimental_GRU_120_24_15_16_24_directdense/model-00030-0.75540-0.85714-0.62804-0.85000.h5
```





5. Bidirectional GRU with 15 units batch size 24:

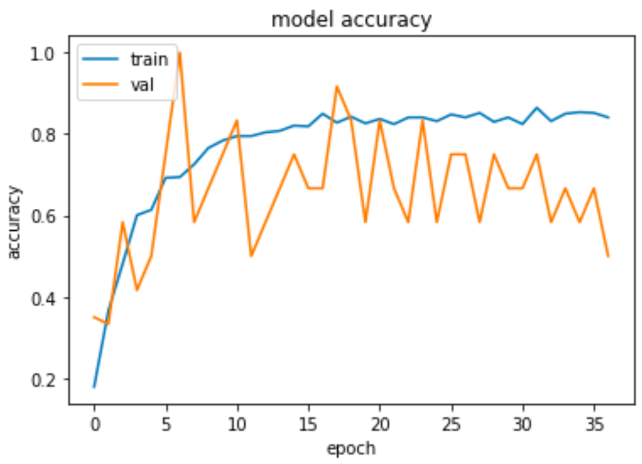


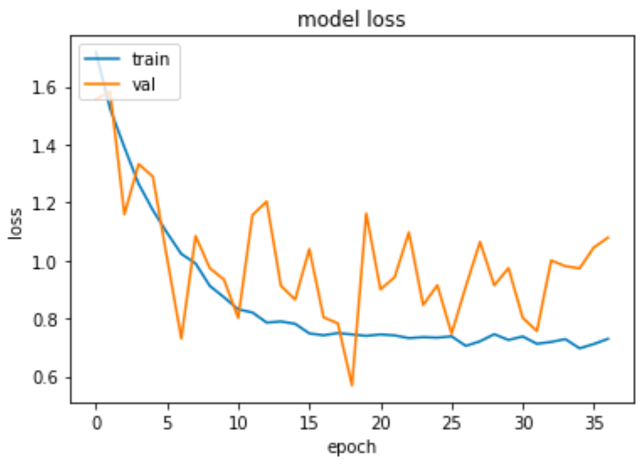


```
28/28 [==============================] - 10s 351ms/step - loss: 0.7915 -
categorical_accuracy: 0.7857 - val_loss: 0.7048 - val_categorical_accuracy: 0.9000

Epoch 00022: val_loss improved from 0.71136 to 0.70482, saving model to
VGG_Experimental_GRUbi_120_24_15_16_24_directdense/
model-00022-0.79151-0.78571-0.70482-0.90000.h5
```

As we can see the model did not have enough predictive power and not hitting high accuracy for training data

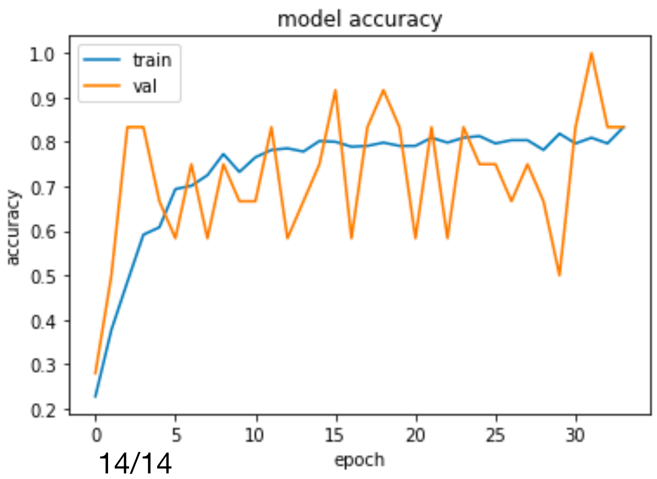
2. GRU 15 units batch size 48:

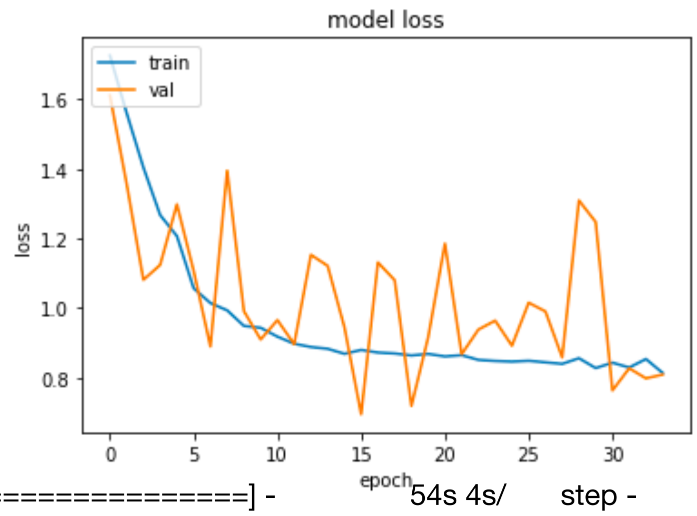




```
14/14 [==============================] - 51s 4s/step - loss: 0.7453 - categorical_accuracy: 0.8425 -
val_loss: 0.5692 - val_categorical_accuracy: 0.8333

Epoch 00019: val_loss improved from 0.73097 to 0.56920, saving model to
VGG_Experimental_GRU_120_48_15_16_48_directdense/model-00019-0.74534-0.84249-0.56920-0.83333.h5
```

6. Bidirectional GRU with 15 units batch size 48:

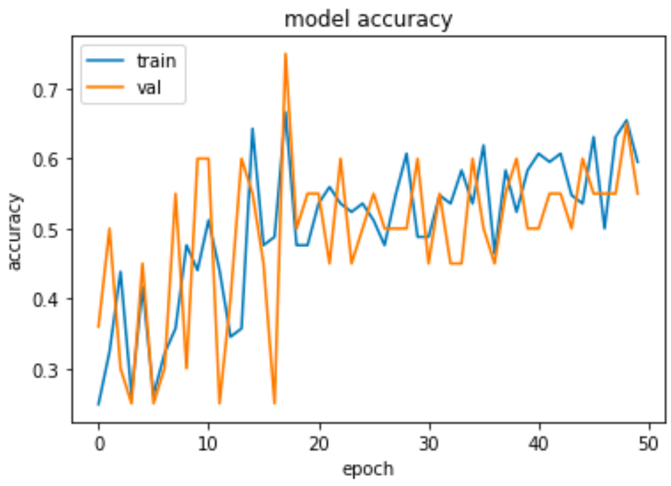


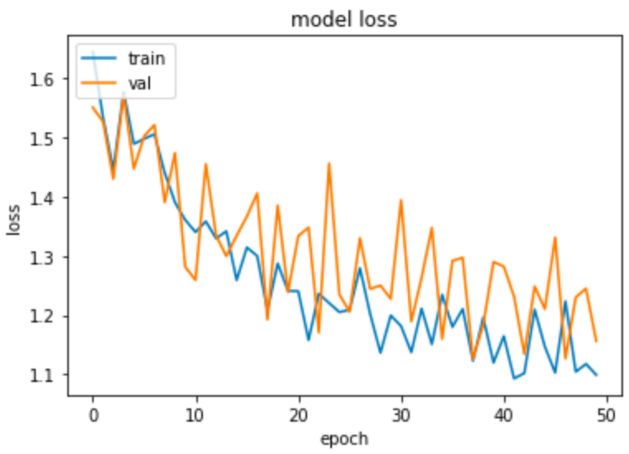


14/14 [==============================] - 54s 4s/ step - loss: 0.8792 - categorical_accuracy: 0.8004 - val_loss: 0.6944 - val_categorical_accuracy: 0.9167

Epoch 00016: val_loss improved from 0.88908 to 0.69444, saving model to VGG_Experimental_GRUbi_120_48_15_16_48_directdense/model-00016-0.87925-0.80037-0.69444-0.91667.h5

3. GRU 8 units with batch size 24:

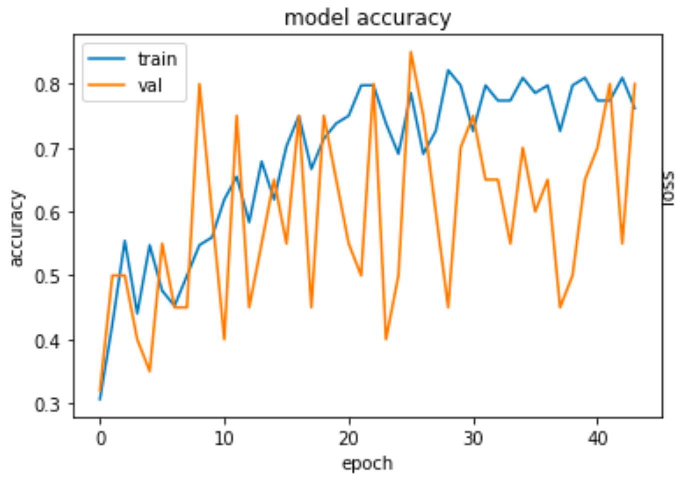


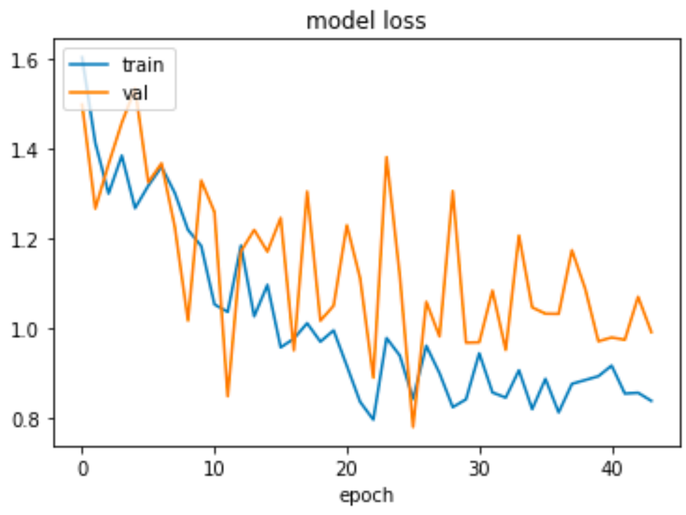


```
28/28 [==============================] - 9s 334ms/step - loss: 1.1229 - categorical_accuracy: 0.5833
- val_loss: 1.1258 - val_categorical_accuracy: 0.5500

Epoch 00038: val_loss improved from 1.15987 to 1.12579, saving model to
VGG_Experimental_GRU_120_24_8_16_24_directdense/model-00038-1.12287-0.58333-1.12579-0.55000.h5
```

Very bad predictive power, 8 units not at all enough

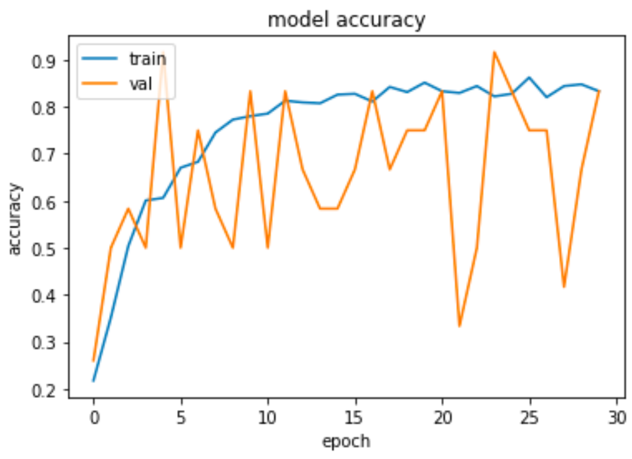
7. Bidirectional GRU with 8 units batch size 24:

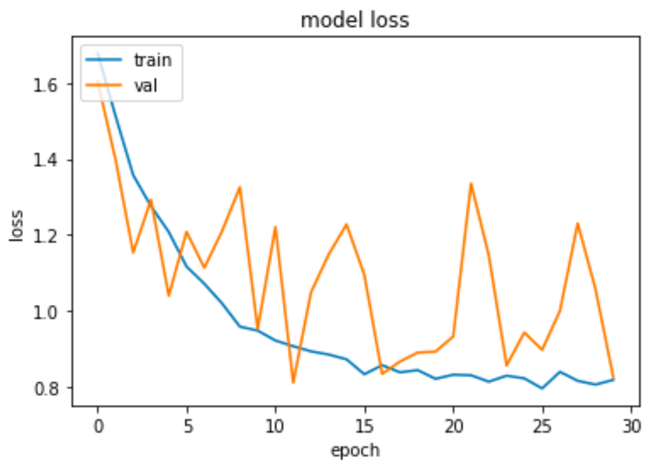




```
28/28 [==============================] - 9s 339ms/step - loss: 0.8438 - categorical_accuracy: 0.7857 - val_loss: 0.7808 - val_categorical_accuracy: 0.8500

Epoch 00026: val_loss improved from 0.84922 to 0.78083, saving model to VGG_Experimental_GRUbi_120_24_8_16_24_directdense/
model-00026-0.84382-0.78571-0.78083-0.85000.h5
```

4. GRU 8 units with batch size 48:

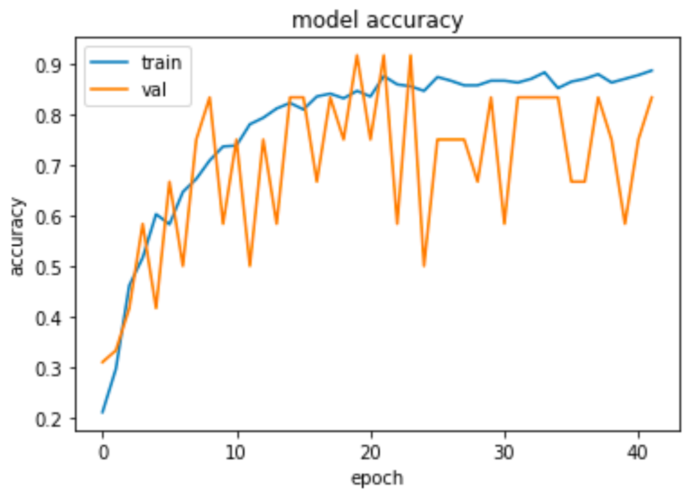


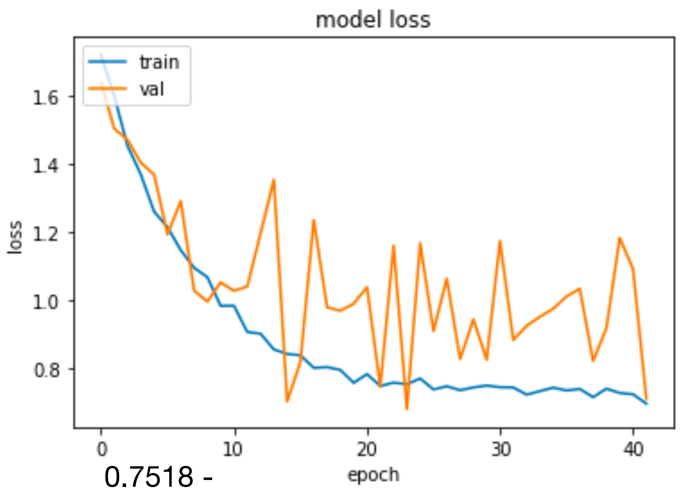


Found Underfitting, more inconsistent than that of batch size of 24.

8. Bidirectional GRU with 8 units batch size 48:

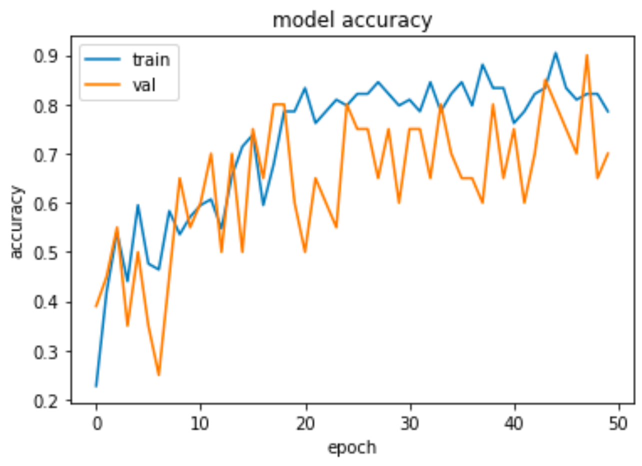
```
14/14 [==============================] - 52s 4s/step - loss:
```

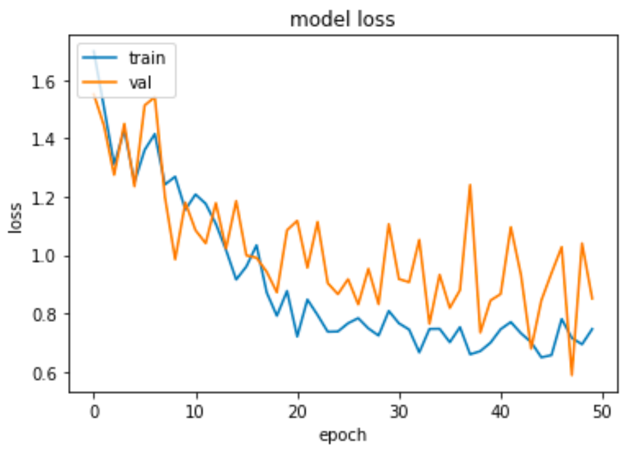




```
0.7518 - categorical_accuracy: 0.8553 - val_loss: 0.6780 -
val_categorical_accuracy: 0.9167

Epoch 00024: val_loss improved from 0.70086 to 0.67795, saving model to
VGG_Experimental_GRUbi_120_48_8_16_48_directdense/
model-00024-0.75180-0.85531-0.67795-0.91667.h5
```

GRU 24 units , batch size 24:

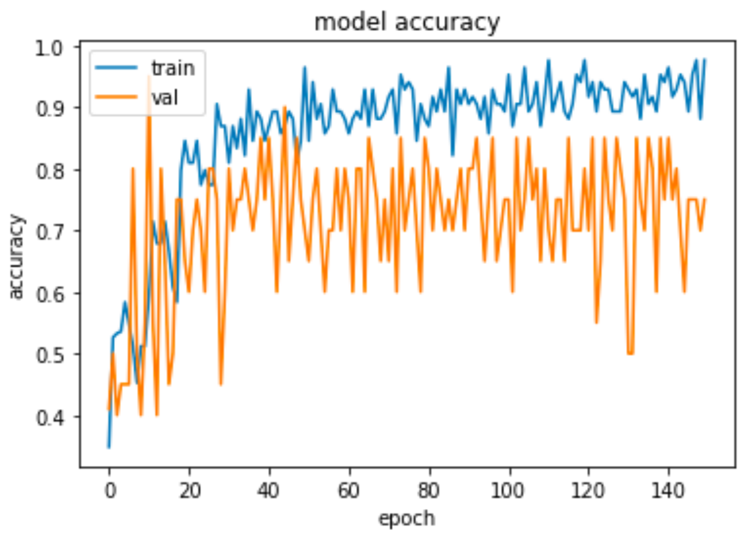


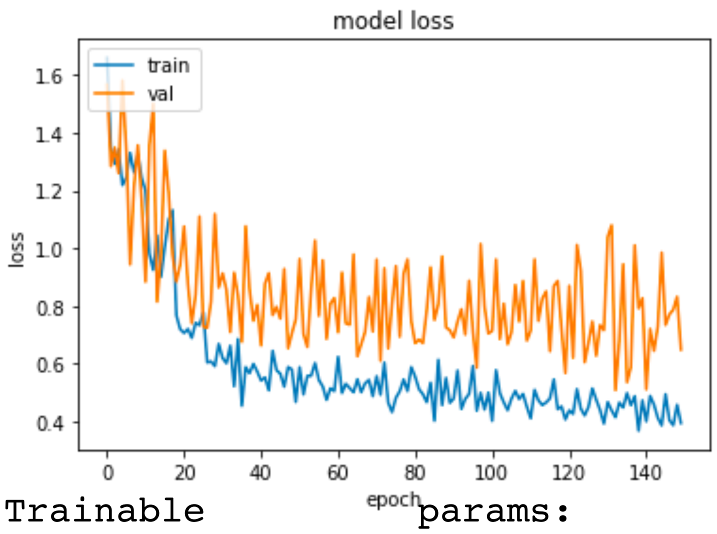


Since the predictive power is less, moving on to bigger network with 40 units, with regularization, and SGD optimiser with momentum 0.9

1. GRU - batch size 24, 24 units:

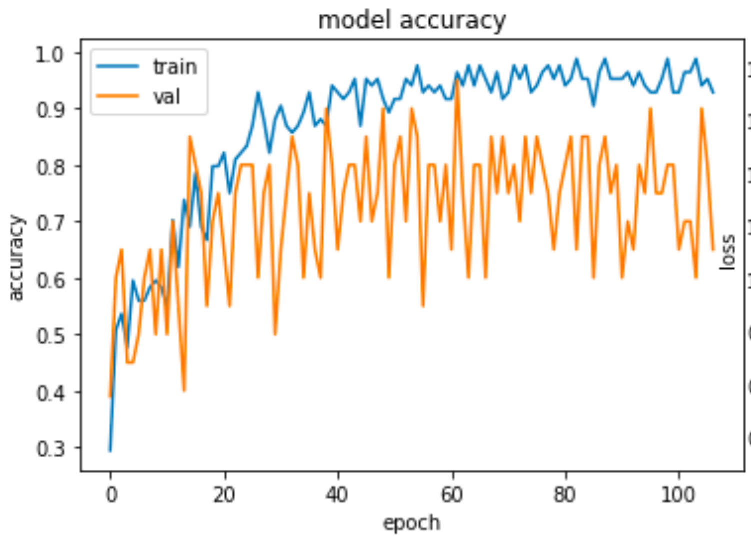
Best performance: model-00133-0.41497-0.92857-0.50838-0.85000.h5 at epoch 133

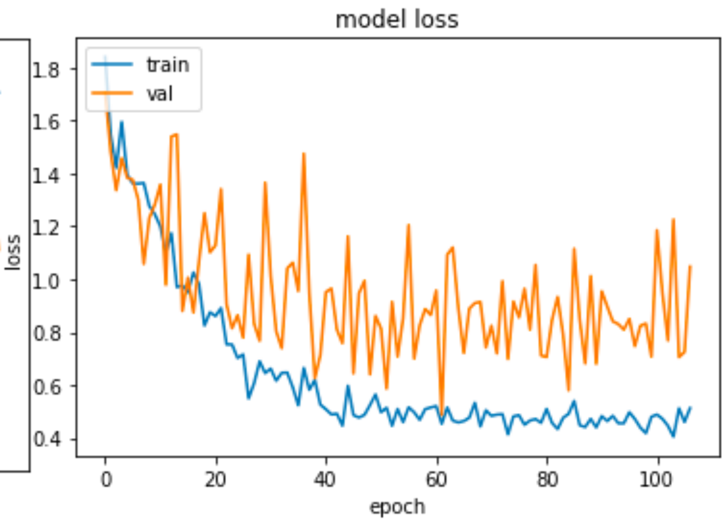




`Trainable params: 558,085`

2. Bidirectional GRU: 24 units

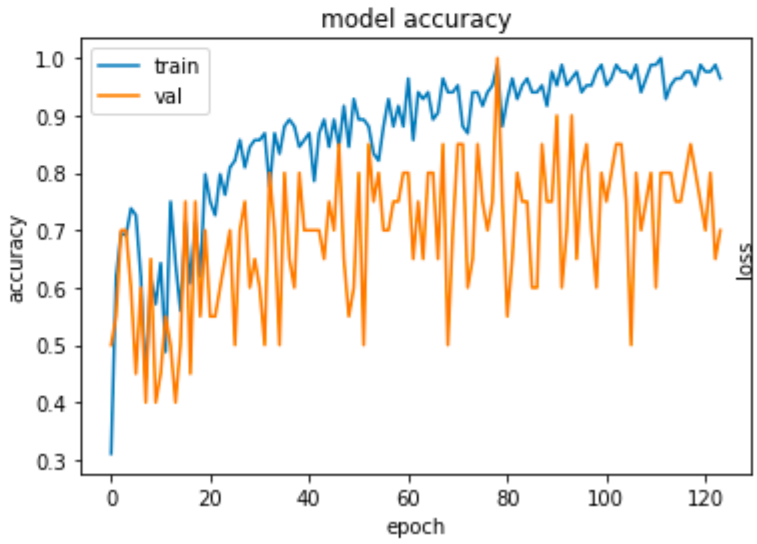


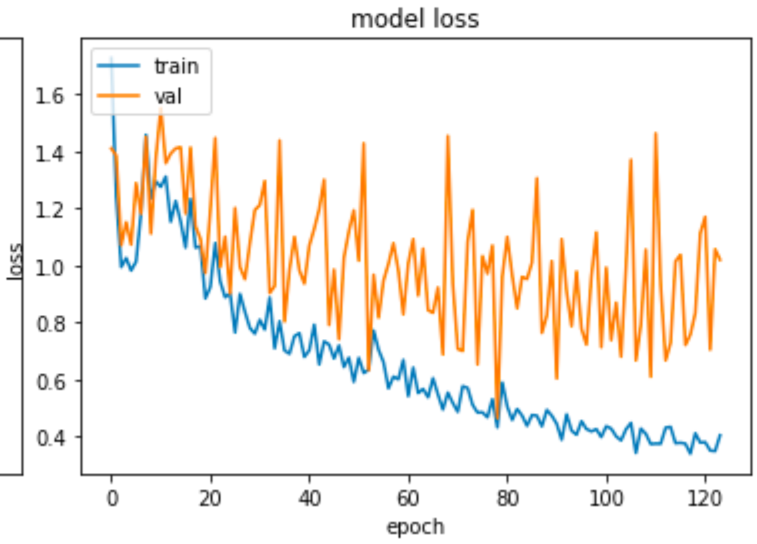


```
28/28 [==============================] - 10s 354ms/step - loss: 0.4532 -
categorical_accuracy: 0.9643 - val_loss: 0.4864 - val_categorical_accuracy: 0.9500

Epoch 00062: val_loss improved from 0.58593 to 0.48644, saving model to
VGG_Experimental_probfinal_GRUbi_120_24_40_16_24_directdense/
model-00062-0.45320-0.96429-0.48644-0.95000.h5
```

`1,116,165`

To further reduce the training loss, Further increasing the predictive power to 64 GRU units, and momentum to 0.99:





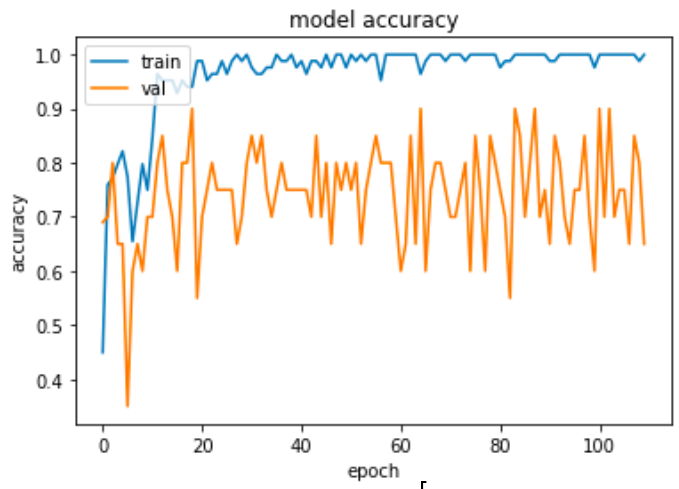
```
Epoch 79/150
28/28 [==============================] - 10s 340ms/step - loss: 0.4310 -
categorical_accuracy: 0.9881 - val_loss: 0.4634 - val_categorical_accuracy: 1.0000

Epoch 00079: val_loss improved from 0.63166 to 0.46344, saving model to
VGG_Experimental_probfinal_56unit_GRU_120_24_64_16_24_directdense/
model-00079-0.43099-0.98810-0.46344-1.00000.h5
```

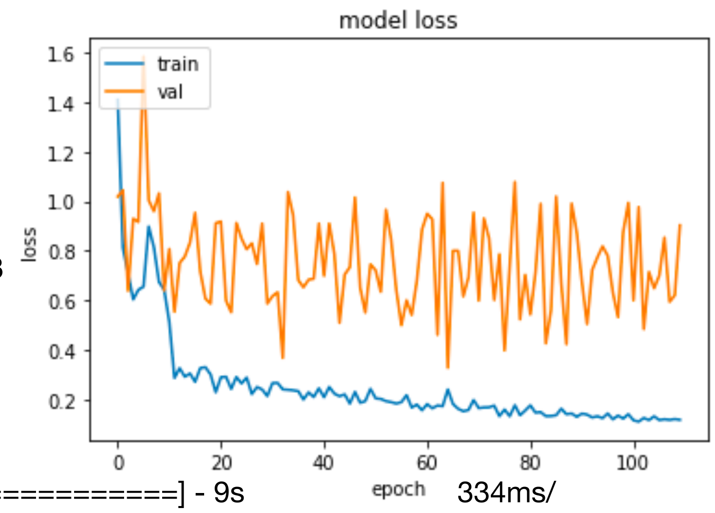
Near perfect accuracy found at 76th epoch.

Even though

Though Bidirectional GRU shows better performance than GRU, it is the problem statement requirement of use of least parameters, choosing to run more powerful GRU network rather than going for Bidirectional GRU makes sense

GRU trying to reduce the power, and changing to Adam:





```
28/28 [==============================] - 9s 334ms/
step - loss: 0.2404 - categorical_accuracy: 0.9643 - val_loss: 0.3277 -
val_categorical_accuracy: 0.9000

Epoch 00065: val_loss improved from 0.36770 to 0.32769, saving model to
VGG_Experimental_probfinal_64unitGRU5xLR_GRU_120_24_30_16_24_directdense/
model-00065-0.24042-0.96429-0.32769-0.90000.h5
```

Tried with multiple momentum values for 0.7, 0.9, 0.99, most of them not showing

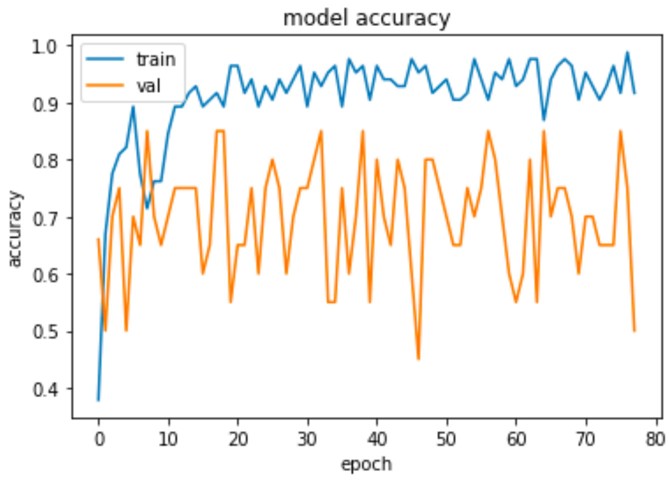
Trials with SGD optimizer shows that SGD might not be the best optimizer for this case, as Adam performed equally better in overfitting the data at 30 GRU units.

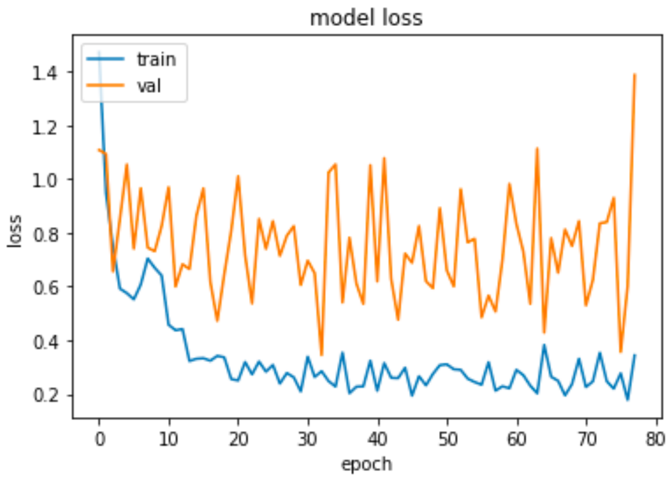
Setting Beta1 for adam at 0.99 and Beta2 at 0.999

Setting very less min cap over learning rate in attempt to smoothen the curve, to 5e-7, over ReduceLROnPlateau

Adding dropouts:

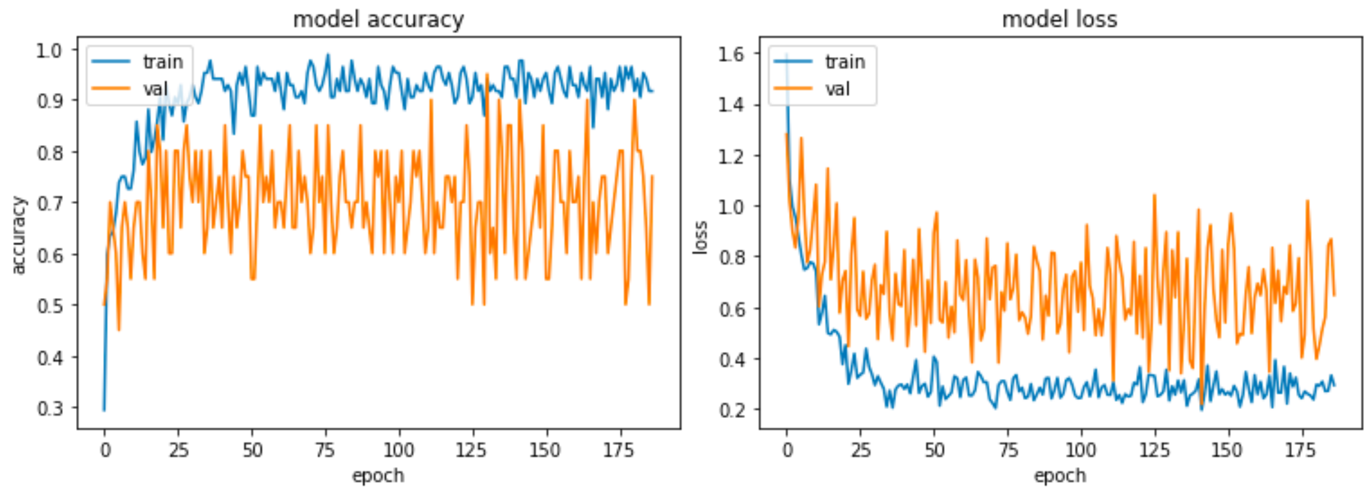
Dropout of 0.2:





```
28/28 [==============================] - 10s 344ms/step - loss: 0.2851 -
categorical_accuracy: 0.9286 - val_loss: 0.3449 - val_categorical_accuracy: 0.8500

Epoch 00033: val_loss improved from 0.47243 to 0.34486, saving model to
VGG_Experimental_BetaTweakerprobfinal_64unitGRU5xLR_GRU_120_24_30_16_0.2_24_directde
nse/model-00033-0.28515-0.92857-0.34486-0.85000.h5
```

Dropout of 0.4:

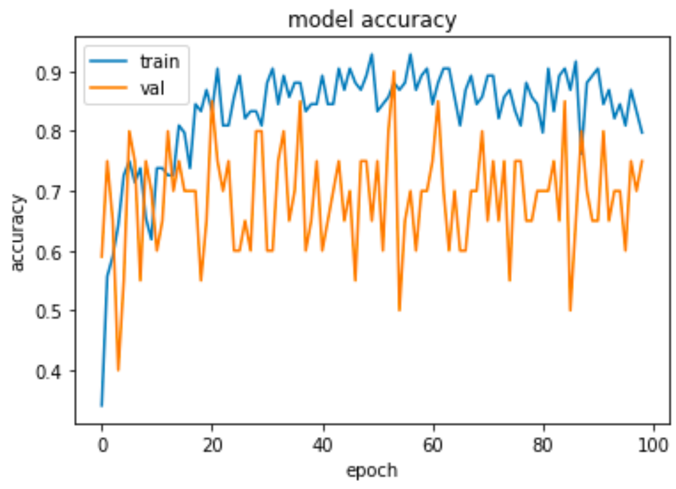


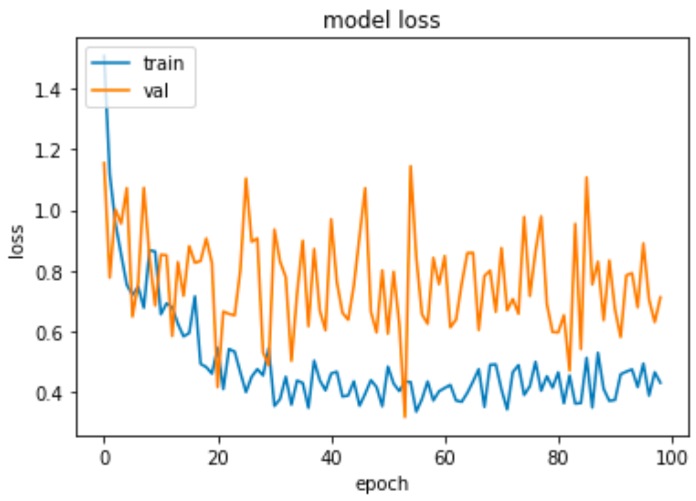
```
Epoch 142/250
28/28 [==============================] - 10s 344ms/step - loss: 0.1959 -
categorical_accuracy: 0.9762 - val_loss: 0.2187 - val_categorical_accuracy: 0.9000

Epoch 00142: val_loss improved from 0.31051 to 0.21875, saving model to
VGG_Experimental_BetaTweakerprobfinal_64unitGRU5xLR_GRU_120_24_30_16_0.4_24_directde
nse/model-00142-0.19594-0.97619-0.21875-0.90000.h5
```

We can see that overfitting is lessened with 0.4 dropout, ie previously, the higher train accuracies were seen around 20th epoch where it went upto 37 epochs to overfit.

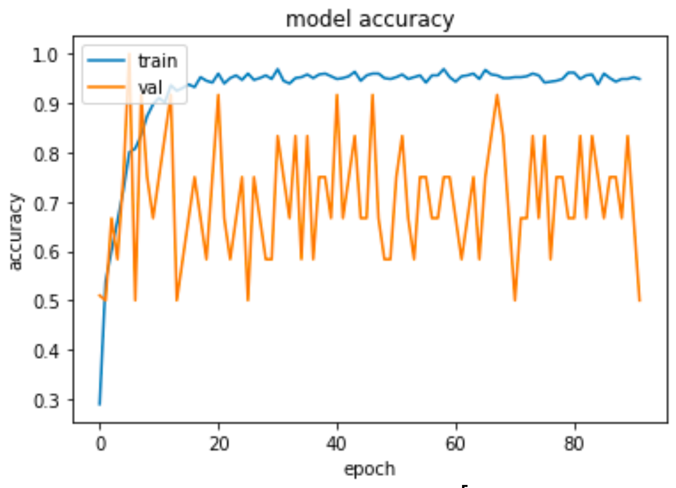
At dropout 0.5: Training loss started taking a hit - making 0.5 too much of a dropout for this exercise

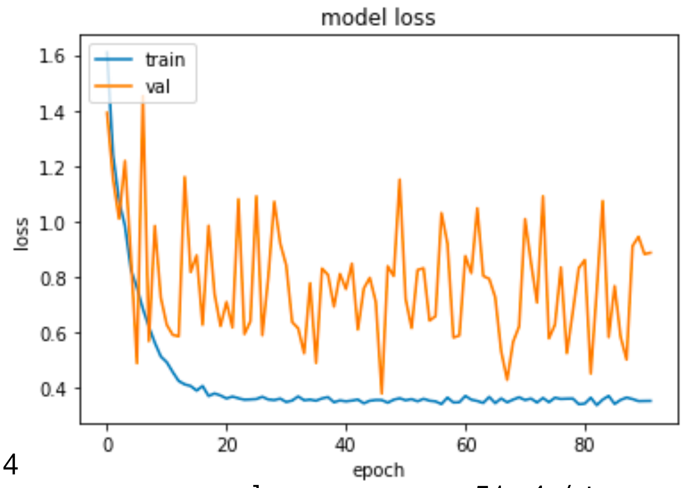
Still, not much accuracy numbers are seen.





Trying the same exercise for GRU 10 units:
With 0.2 dropout, Adam optimizer with above mentioned configuration:





```
14/14
[==============================] - 51s 4s/step -
loss: 0.3571 - categorical_accuracy: 0.9597 - val_loss: 0.3806 - val_categorical_accuracy: 0.9167

Epoch 00047: val_loss improved from 0.48946 to 0.38056, saving model to
VGG_Experimental_BetaTweakerprobfinal_10unit_GRU_120_48_10_16_0.2_48_directdense/
model-00047-0.35715-0.95971-0.38056-0.91667.h
```

It can be understood that there is very less predictive power for GRU with 10Units as the best training loss is 0.357

Trying the same exercise for GRU 48 units:

Overfitting

Trying the same exercise for GRU 24 units:

Slightly underfitting

Since the best model we have got is of 64 GRU units, which is 69MB is size, going with Bidirectional GRU of 24 units, with even lesser parameters:

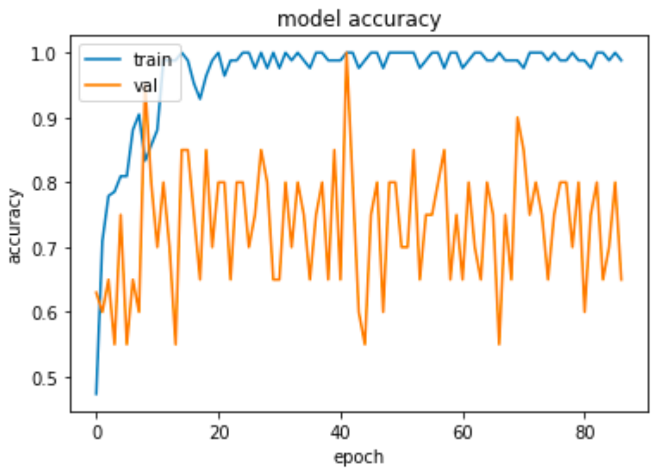
```
===================================================
Evaluating models for type_rnn: GRUbi dropout: 0.4 __rnn_unit: 24 and image_count: 16
image_size: 120  and batch_size: 24
===================================================
2020-08-24 16:49:19
GRUbi  is type of rnn
VGG_Experimental_BetaTweakerprobfinal_6448unit_GRUbi_120_24_24_16_0.4  is
model_name_prefix
24  gru units taken
# training sequences = 663
# validation sequences = 100
# epochs = 150
setting dropout: 0.4
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         (None, 16, 120, 120, 3)   0
_________________________________________________________________
time_distributed_1 (TimeDist (None, 16, 3, 3, 512)     14714688
_________________________________________________________________
time_distributed_2 (TimeDist (None, 16, 4608)          0
_________________________________________________________________
bidirectional_1 (Bidirection (None, 48)                667152
_________________________________________________________________
output (Dense)               (None, 5)                 245
=================================================================
Total params: 15,382,085
Trainable params: 667,397
Non-trainable params: 14,714,688
```

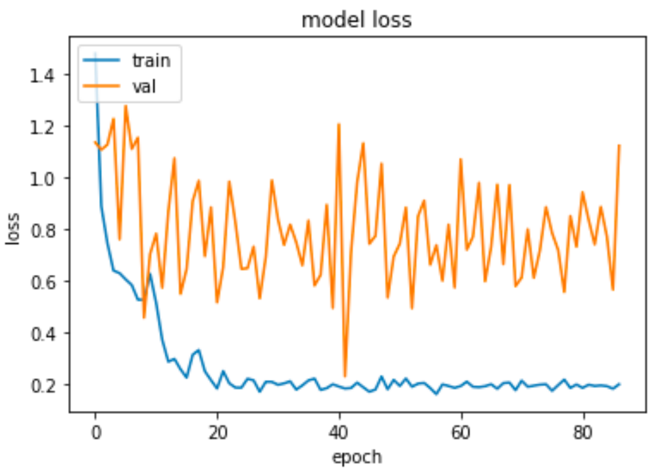
Though better models are obtained with merely 130K parameters, this particular model is heavily regularized, and also model size is humble enough of 69MB

```
Epoch 00041: ReduceLROnPlateau reducing learning rate to 6.553600542247295e-07.
Epoch 42/150
28/28 [==============================] - 10s 351ms/step - loss: 0.1819 -
categorical_accuracy: 1.0000 - val_loss: 0.2289 - val_categorical_accuracy: 1.0000

Epoch 00042: val_loss improved from 0.45622 to 0.22892, saving model to
VGG_Experimental_BetaTweakerprobfinal_6448unit_GRUbi_120_24_24_16_0.4_24_directdense/
model-00042-0.18195-1.00000-0.22892-1.00000.h5
```

Train Accuracy : 100%
Train loss : 0.18195
Validation Accuracy : 100%
Validation loss : 0.22892

Because of regularization, we can expect the model to generalize well. Also, the Bidirectional aspect of model intuitively makes sense to utilize temporal information





Note: Submitting the near perfect model of GRU of 64 units as final model.

It is seen that with minimal amounts of learning rate, regularization, momentum adjustment, Beta adjustments of Adam optimizers, The inconsistency of validation curve is noted.

We have also tried working with 300:100 samples of training and validation data. Similar behaviour is found, which implies that this is because of type of training data.

Submitting the model as final submission where most validation curve consistency, as well as good accuracy is found:

```
Epoch 79/150
28/28 [==============================] - 10s 340ms/step - loss: 0.4310 -
categorical_accuracy: 0.9881 - val_loss: 0.4634 - val_categorical_accuracy: 1.0000

Epoch 00079: val_loss improved from 0.63166 to 0.46344, saving model to
VGG_Experimental_probfinal_56unit_GRU_120_24_64_16_24_directdense/
model-00079-0.43099-0.98810-0.46344-1.00000.h5
```

Near perfect accuracy found at 76th epoch.