Gesture Recognition Model building and selection approach

# Objective:

We are provided with Train and validation dataset containing 5 gestures. We have to build a model which will detect the hand gestures.

# Approach:

1. Create a generator to load the dataset in batches in memory. We have also added code to read data to partially read the data.
2. On a smaller dataset perform experiments with different model architecture and learning rates. For this exercise we have selected adam optimizer only.
3. Select final model and train on whole dataset and evaluate model on validation dataset.

Following sections will provide more details on each experimental model:

# Experiment 1: Time distributed CNN + GRU (ARCH 1)

## Model architecture:

ANN (5 Neurons + Softmax activation)

Dense Layer/s

ANN

GRU

Time distributed (Flatten)

Time distributed (Batch Normalization +

MaxPool2D)

Time distributed (Inception V3 + pre-trained imagenet)

…….

…….

…….

(6)

(5)

(4)

(3)

(2)

(1)

*Total Parameters:*

*Total params: 32,305,957*

*Trainable params: 32,267,429*

*Non-trainable params: 38,528*

*Code Snippet:*

*model\_3 = Sequential()*

*model\_3.add(TimeDistributed(cnn\_model,input\_shape=(x,y,z,3)))*

*model\_3.add(TimeDistributed(BatchNormalization()))*

*model\_3.add(TimeDistributed(MaxPooling2D((2, 2))))*

*model\_3.add(TimeDistributed(Flatten()))*

*model\_3.add(GRU(1024))*

*model\_3.add(Dropout(0.5))*

*model\_3.add(Dense(1024,activation='relu'))*

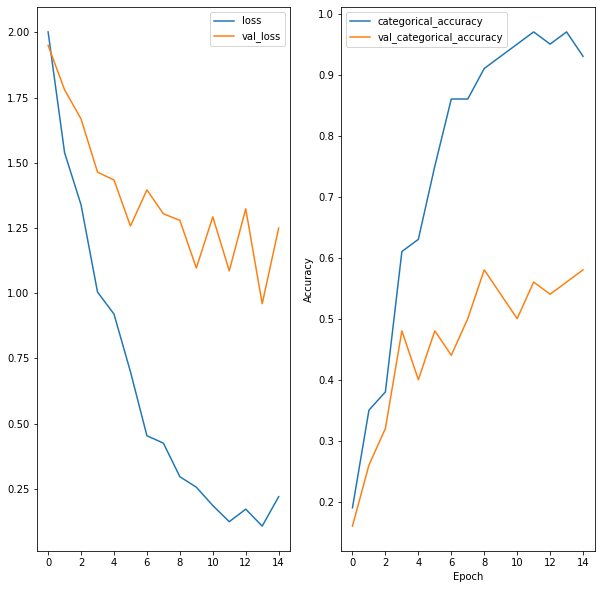
*model\_3.add(Dropout(0.5))*

*model\_3.add(Dense(5, activation='softmax'))*

## Plots:

The loss and accuracy trend on training (100 sequences) and validation (50 sequences).

Following plot is from lr=0.001 and epoch=15.



## Inference:

We tried various learning rates but there were no improvements. The model was overfitting the training data. So we changed the architecture in next experiment

# Experiment 2: Time distributed CNN + GRU (ARCH 2)

## Model architecture:

We reduced the GRU cells (component (4)) and increased one dense layer (component (5)) from our [previous model](#_Model_architecture:).

*Total Parameters:*

*Total params: 22,221,397*

*Trainable params: 22,182,869*

*Non-trainable params: 38,528*

*Code Snippet:*

*model\_2 = Sequential()*

*model\_2.add(TimeDistributed(cnn\_model,input\_shape=(x,y,z,3)))*

*model\_2.add(TimeDistributed(BatchNormalization()))*

*model\_2.add(TimeDistributed(MaxPooling2D((2, 2))))*

*model\_2.add(TimeDistributed(Flatten()))*

*model\_2.add(GRU(64))*

*model\_2.add(Dropout(0.5))*

*model\_2.add(Dense(64,activation='relu'))*

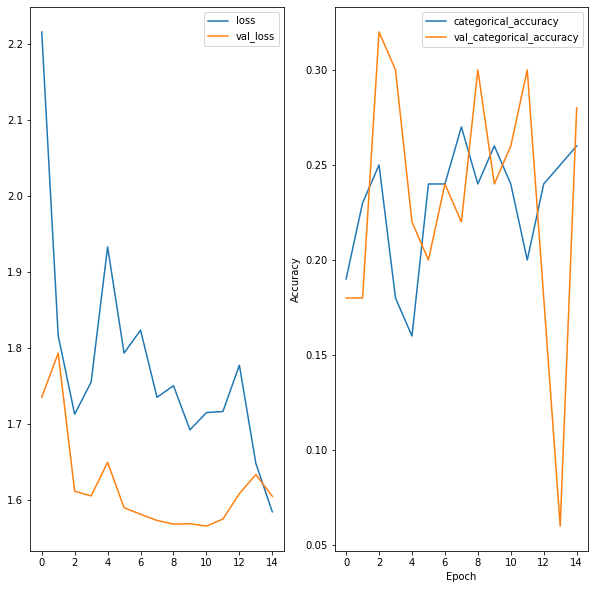
*model\_2.add(Dropout(0.5))*

*model\_2.add(Dense(8,activation='relu'))*

*model\_2.add(Dropout(0.5))*

*model\_2.add(Dense(5, activation='softmax'))*

## Plots:



## Inference:

From the plot the model behavior is almost same on training and validation data. The loss is reducing while training. But we are not getting more accuracy. This could be because we have less GRU cells and less neuron in dense layers.

# Experiment 3: Time distributed CNN + GRU (ARCH 3)

## Model architecture:

To achieve better accuracy we increased GRU cells (component (4)) and increased neurons in dense layer (component (5)) from our [previous model](#_Model_architecture:_1).

*Code Snippet:*

*model\_1 = Sequential()*

*model\_1.add(TimeDistributed(cnn\_model,input\_shape=(x,y,z,3)))*

*model\_1.add(TimeDistributed(BatchNormalization()))*

*model\_1.add(TimeDistributed(MaxPooling2D((2, 2))))*

*model\_1.add(TimeDistributed(Flatten()))*

*model\_1.add(GRU(256))*

*model\_1.add(Dropout(0.5))*

*model\_1.add(Dense(256,activation='relu'))*

*model\_1.add(Dropout(0.5))*

*model\_1.add(Dense(128,activation='relu'))*

*model\_1.add(Dropout(0.5))*

*model\_1.add(Dense(5, activation='softmax'))*

*model\_2.add(Dense(5, activation='softmax'))*

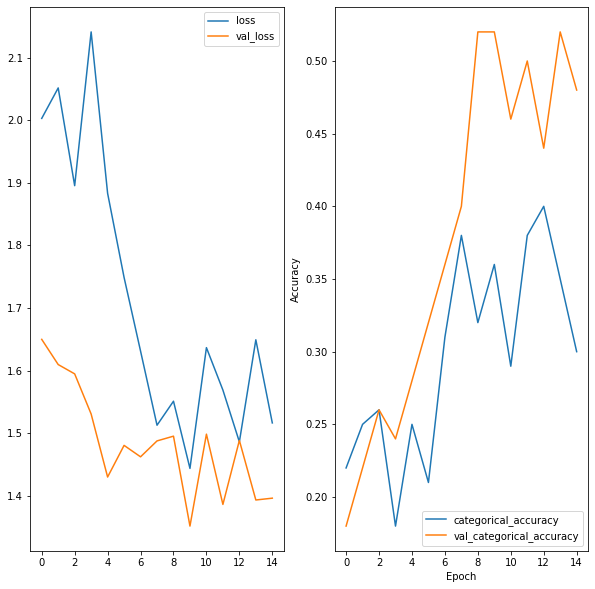
*Total Parameters:*

*Total params: 23,680,549*

*Trainable params: 23,642,021*

*Non-trainable params: 38,528*

## Plots:



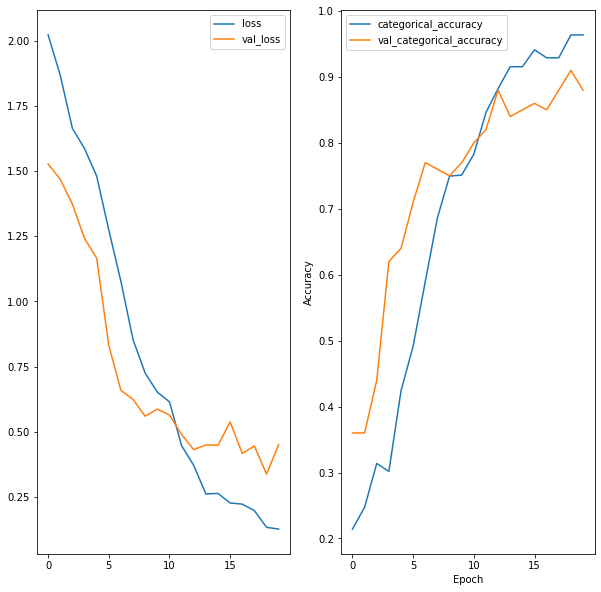
## Inference:

From the plot the model behavior is almost same on training and validation data. The loss is reducing and also the accuracy is increasing.

This model architecture can be used train on the whole dataset. We trained it for 15 epochs and were able to get train accuracy=97% and validation accuracy = 95.

Further using this model for evaluation and prediction on validation confirmed it. This model code is available in the **notebook uploaded.**

The final training plot is as follows:



# Decision Matrix:

## CNN + GRU:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| 1 | Time distributed CNN + GRU  (ARCH 1) | Overfitting on training data. | Epoch=15, dataset = 100 sequence  Loss = 0.21 Accuracy = 0.93  Validation loss = 1.24  Validation Accuracy = 0.58  (refer [plot](#_Plots:))  Total params: 32,305,957  The model is overfitting on the training data and no change happened on changing leaning rate. So changed the architecture |
| 2 | Time distributed CNN + GRU  (ARCH 2) | Loss and accuracy trend is same but value of accuracy needs improvement. | Epoch=15, dataset = 100 sequence  Los= 1.58 Accuracy = 0.26  Validation loss = 1.6  Validation Accuracy = 0.28  (refer [plot](#_Plots:_1))  Total params: 22,221,397  The model behaving same on training and validation data. Accuracy and loss has not reduced. Need to increase neurons. |
| 3 | Time distributed CNN + GRU  (ARCH 3) Best Model\* | Improvement in accuracy | Epoch=15, dataset = 100 sequence  Loss=1.51 Accuracy = 0.30  Validation loss = 1.39  Validation Accuracy = 0.48 (refer [plot](#_Plots:_2))  Epoch=20, dataset = 663 sequence  Loss=0.133 Accuracy = 0.96  Validation loss = 0.  Validation Accuracy = 0.95  Total params: 23,680,549 |

## CONV3D:

We also tried to build gesture recognition using Conv3D layers. Following are the details

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Experiment no** | **Model Parameters** | **Training Accuracy** | **Validation Accuracy** | **Epochs** | **Comments** |
| 1 | Total params: 4,999,557  Trainable params: 4,999,109  Non-trainable params: 448 | 75 | 33 | 10 | Simple model - no dropouts.  Overfit.  Image size - 100 \*100.  No Crop.  Tested on 100 Train and validation set.  Batch size -50  Sample frames - 18 |
| 2 | Total params: 892,101  Trainable params: 891,237  Non-trainable params: 864 | 78.7 | 20 | 15 | Dropouts -0.5 and 2 dense layers(64). Overfit. Not good.  Tested on 100 Train and validation set.  Batch size -100  Sample frames - 13 |
| 3 | Total params: 1,313,797  Trainable params: 1,313,093  Non-trainable params: 704 | 43 | 22 | 15 | Reduced one dense layer.  Underfit model.  Tested on 100 train and validation set.  Batch size - 50 |
| 4 | Total params: 5,000,069  Trainable params: 4,999,365  Non-trainable params: 704 | 96.7 | 16 | 20 | Bad overfit model.  Tested on 100 train and validation set. Batch size - 32.  Dense layer tried with 128 instead of 64. |
| 5 | Total params: 5,000,069  Trainable params: 4,999,365  Non-trainable params: 704 | 97 | 29 | 20 | Till now Batch normalisation was done on the inputs. In this model tried batch normalisation after applying activation function.  Overfit Model. |
| 6 | Total params: 4,999,557  Trainable params: 4,999,109  Non-trainable params: 448 **(Best Model)** | 75 | 74.2 | 50 | Made changes on experiment 1 - Dropouts after dense layer -0.25.  Learning rate taken as 0.0001 for Adam optimiser.  Batch size – 32  Batch normalization before activation function.  Sample frames - 18  Size - 100\*100  **Code for this model is submitted as part of this assignment** |

## Model Comparison

|  |  |  |
| --- | --- | --- |
| **Best Models from both Arch** | **Accuracy** | **Total Params** |
| Experiment3 Model 3 CNN + GRU | Train = 96 Val = 89 | 23,680,549 (File size ~ 270MB) |
| Experiment6 Conv3D | Train = 75 Val = 74.2 | 4,999,557 |

# Model Selection:

Conv3D model is the best model to be used in production with smart-tv as it is simple with 5 times less parameters and easy to understand architecture (This is in accordance with Occams Razor principle as well).

Note \*1: For Notebook with CNN+GRU please contact us [deb.bis220@gmail.com](mailto:deb.bis220@gmail.com) or [sophyiyacit@gmail.com](mailto:sophyiyacit@gmail.com). We understood that only one final model needs to be submitted so not all best models.