Hand Gesture Recognition Using CNN Based Models

- Rajagopal Shenoy G
- Anindya Sudhir

Problem Statement

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We want to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

• Thumbs up	: Increase the volume.
• Thumbs down	: Decrease the volume.
• Left swipe	: 'Jump' backwards 10 seconds.
• Right swipe	: 'Jump' forward 10 seconds.
• Stop	: Pause the movie.

Objective

Train different models on the 'train' folder to predict the action performed in each sequence and compare its performance on the validation data provided in 'val' folder.

Our Approach

1. Five types of NN models are trained:

1. 3D Convolutional Neural Networks (Conv3D)
2. LSTM model based on CNN + RNN architecture
3. GRU model based on CNN + RNN architecture
4. RESNET50 Transfer Learning Model
5. VGG16 Transfer Learning Model

2. Data Pre-processing using the Data Generator

In the generator function, we are going to pre-process the images as we have images of 2 different dimensions (360*360 and 120*160) as well as create a batch of video frames.
The generator function is capable to do the following:

1. **Batch size**: Split the data set into batches as per the batch size decided
2. **No. of Frames**: Select a smaller set of frames from the given 30 frames (i.e., it can reduce the number of images and hence achieve smaller frame rate)
3. **Resizing**: Resize the images (i.e., change the image height and width)
4. **Normalize** the RGB values of the images (is done to get rid of distortions caused by lights and shadows in an image). All RGB channels are normalized with the factor of 255 as we achieved good accuracy with that. Hence, there was no need to experiment with separate normalization for each channel.
5. If augmentation is required, then accordingly create images with **Gaussian Noise.**

3. Model Trainer Function

A user defined function to fit and train the model on training data and validate on the validation data is written.
This calls the generator function to pass the data to the model as per the batch size determined.
This function also defines three callback functions while training the model:

a. **Callback function to reduce learning rate upon reaching a plateau**
b. **Callback function for Early Stopping** (This feature enables us to go for higher number of epochs (i.e., 100 to be precise) as we know that if the model is not performing any better, Early Stopping will kick in and cancel the further epochs)
c. **Callback function to save all the models**

4. Function to plot Model Accuracy

A user defined function to show the loss and accuracy of the model on training and validation data in every epoch is written.

5. Determining the Batch Size, Number of Frames and Image Size

Time taken and the memory needed to train the model is greatly affected by the batch size, image size and number of frames. Let us chose some values judiciously.
First a sample model (model_Conv3D_Test) is written purpose of which is just to determine these values.

Batch Size:
- We experimented with different batch size till we hit the OOM error of GPU
- We found that batch size around **50** is a good value. It can be increased even up to 100 if we do not decide to use the augmented images.

Image Size:
We experimented with different image sizes - 100x100, 120x120 and 160x160 and found **120x120** is good (it balances out the time, memory and at the same time does not result in loss of accuracy)

Number of Frames:
We went ahead with one thought: Even one single image is capable enough to determine whether it is a Thumbs Up, Thumbs Down or Stop. For identifying Left and Right Swipes, we need at the most two images. This means we can manage with a smaller number of frames instead of all 30. So, we chose **10** as the number of frames (This analysis of ours proved right as we later on when we check the accuracy of the models). This also helps us to tackle overfitting.

6. Determining the Activation Function

- It is clear that the final **output layer** should use **softmax** as the activation function to determine the 5 classes.
- But for the inner layers, a decision has to be made on which activation function suits the best. We came across this stackoverflow thread: https://stats.stackexchange.com/questions/444923/activation-function-between-lstm-layers
- It says "Given that ReLUs can have quite large outputs, they have traditionally been regarded as inappropriate for use with LSTMs"
- Hence, we use **ReLU** for **Conv3D models** and **tanh** for **LSTM/GRU models** as activation functions *(We also verified this with running the models with both activation functions, but this code is not included in the notebook submitted to not to make the notebook too cluttered)*

7. NN Architecture development and training

- We started with the Conv3D models first
- We started with models with 4 Conv3D layers with no padding, with Batch Normalization and Max Pooling at the end of each layer. They are then flattened. Then two dense layers along with Batch Normalization and DropOuts are added. Finally a softmax layer with 5 dense neurons is added.
- We started with a smaller number of epochs (20) in the beginning (from Model 1 to 6) till we get all the parameters we need. Then we will switch to higher number of epochs (100) (Model 7 onwards) when we seriously start to train with all hyperparameters decided.
- Following steps helped in creating the models and choosing the winner:

    1. First step was to choose the correct kernel size between (3,3,3) and (2,2,2). Model 1 and Model 2 were used for this and we found out that Model 2 was better than Model 1 in terms of accuracy which meant we chose **(2,2,2) as the kernel size**.

    2. Second step was to check how DropOuts affect the performance of the model. For this, we created Model 3 in which we added DropuOuts after every convolution layer. We found out that **adding DropOuts at the end of every convolution layer is a bad idea**.

    3. Third step was to fix the number of neurons in the dense layers. We experimented with 128 neurons (Model 4) and 64 neurons (Model 5) and found out that both models fared equally bad (overfitted). So we went with Model 4 i.e. **dense layers with 128 neurons**.

    4. Fourth step was to decide between the Adam and the SGD Optimizers. All the models run so far was on Adam Optimizer. Model 6 was run using SGD Optimizer. We found out that **Adam Optimizer is the better choice** as it performed a bit better.

    5. Now that all the hyperparameters are decided, time to get serious and start building the models with higher number of epochs. In the Fifth step, based on the hyperparameters chosen above, Model 7 is created and is run for 100 epochs (or till we hit early stopping). This model performed excellent and gave us an **accuracy of 93.8% on training and 89% on validation data**.

    6. Sixth step was to rerun the model 7 but this time with augmentation enabled. This is called Model 8. We saw that augmentation deteriorated the model performance and hence we chose to **not use the image augmentation**.

       Now that the Model 7 has been finalized from Conv3D Models. Next steps are to train LSTM and GRU models and see how they perform.

    7. Seventh step was to create Model 9 which is a CNN LSTM model. We see that Model 9 also has also performed well (94% training and 84% accuracy)

    8. Eighth step was to create Model 10 which is a CNN GRU model. As expected, performance of this model is close to the LSTM model but is not as good as the LSTM model.

    9. Ninth step was to create Model 11 which is a Transfer Learning model based on RESNET50 architecture. It performed really bad with severe overfitting.

10. Tenth step was to create Model 12 which is a Transfer Learning model based on VGG16 architecture. It started to overfit in the later epochs and hence we dropped this model from the race.

- For detailed information please refer below table. Winner is highlighted in green.

No.	Model Name	Model	Result	Decision + Explanation
0	Conv3D_Test	Conv3D	-	Batch size of 50, number of frames as 10 and image size as 120x120 are chosen
1	Conv3D_1	Conv3D with (3,3,3) filter size	Accuracy - Training: 97% Validation: 76%	Filter size of (2,2,2) is chosen over (3,3,3) as it performs better (lesser overfit) and also means less parameters to train
2	Conv3D_2	Conv3D with (2,2,2) filter size	Accuracy - Training: 99% Validation: 83%	
3	Conv3D_3	Conv3D with DropOut after each conv layer	Accuracy - Training: 72% Validation: 28%	Model performance is bad (severe overfit). So do not use DropOut after every conv layer.
4	Conv3D_4	Conv3D with 128 Dense Neurons	Accuracy - Training: 99% Validation: 85%	Both models performed equally bad. We went ahead with 128 neurons in each dense layer
5	Conv3D_5	Conv3D with 64 Dense Neurons	Accuracy - Training: 99% Validation: 83%	
6	Conv3D_6	Conv3D with SGD Optimizer	Accuracy - Training: 99% Validation: 80%	Previous models had Adam Optimizer. Adam Optimizer is chosen over the SGD as it has performed slightly better
7	Conv3D_7 (Winner)	Conv3D with all the learnings of previous models but with no augmentation	Accuracy - Training: 93.8% Validation: 89%	Image Augmentation did not improve the accuracy, instead it caused overfitting. Hence augmentation was dropped from the dataset. This also means less amount of data to train on making it faster. **So, Model 7 is our final chosen model**
8	Conv3D_8	Conv3D with all the learnings of previous models but with augmentation (Gaussian Noise)	Accuracy - Training: 93% Validation: 78%	
9	LSTM_9	ConvLSTM	Accuracy - Training: 94% Validation: 84%	Performance is quite good as Model 7. But Model 7 is still a better model
10	GRU_10	ConvGRU	Accuracy - Training: 89% Validation: 83%	Performance comes close second to that of LSTM model
11	Resnet50_11	ResNet – LSTM Transfer Learning Model	Accuracy - Training: 91% Validation: 28%	Badly overfitting and will not be pursued further
12	Vgg16_12	VGG16 – LSTM Transfer Learning Model	Accuracy - Training: 98% Validation: 82%	Model started overfitting in the later epochs
Final Model	Conv3D_7 (highlighted in green above)	Conv3D Model	Accuracy - Training: 93.8% Validation: 89%	This model is giving fantastic accuracy and is not overfitting at all

8. Conclusion

- Model 7 (Conv3D) model, Model 9 (LSTM) and Model 10 (GRU) have performed really well.
- But Model 7 (Conv3D) is the clear winner as it has given best accuracy (93.8% - training, 89% - validation) with no overfitting.
- We chose Model 7 (Conv3D) as our final model.
- Model 7 summary:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv3d_77 (Conv3D)           (None, 10, 120, 120, 16)  400       
_________________________________________________________________
activation_287 (Activation)  (None, 10, 120, 120, 16)  0         
_________________________________________________________________
batch_normalization_112 (Bat (None, 10, 120, 120, 16)  64        
_________________________________________________________________
max_pooling3d_77 (MaxPooling (None, 5, 60, 60, 16)     0         
_________________________________________________________________
conv3d_78 (Conv3D)           (None, 5, 60, 60, 32)     4128      
_________________________________________________________________
activation_288 (Activation)  (None, 5, 60, 60, 32)     0         
_________________________________________________________________
batch_normalization_113 (Bat (None, 5, 60, 60, 32)     128       
_________________________________________________________________
max_pooling3d_78 (MaxPooling (None, 3, 30, 30, 32)     0         
_________________________________________________________________
conv3d_79 (Conv3D)           (None, 3, 30, 30, 64)     16448     
_________________________________________________________________
activation_289 (Activation)  (None, 3, 30, 30, 64)     0         
_________________________________________________________________
batch_normalization_114 (Bat (None, 3, 30, 30, 64)     256       
_________________________________________________________________
max_pooling3d_79 (MaxPooling (None, 2, 15, 15, 64)     0         
_________________________________________________________________
conv3d_80 (Conv3D)           (None, 2, 15, 15, 128)    65664     
_________________________________________________________________
activation_290 (Activation)  (None, 2, 15, 15, 128)    0         
_________________________________________________________________
batch_normalization_115 (Bat (None, 2, 15, 15, 128)    512       
_________________________________________________________________
max_pooling3d_80 (MaxPooling (None, 1, 8, 8, 128)      0         
_________________________________________________________________
dropout_60 (Dropout)         (None, 1, 8, 8, 128)      0         
_________________________________________________________________
flatten_19 (Flatten)         (None, 8192)              0         
_________________________________________________________________
dense_51 (Dense)             (None, 128)               1048704   
_________________________________________________________________
activation_291 (Activation)  (None, 128)               0         
_________________________________________________________________
batch_normalization_116 (Bat (None, 128)               512       
_________________________________________________________________
dropout_61 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_52 (Dense)             (None, 128)               16512     
_________________________________________________________________
activation_292 (Activation)  (None, 128)               0         
_________________________________________________________________
batch_normalization_117 (Bat (None, 128)               512       
_________________________________________________________________
dropout_62 (Dropout)         (None, 128)               0         
_________________________________________________________________
dense_53 (Dense)             (None, 5)                 645       
_________________________________________________________________
activation_293 (Activation)  (None, 5)                 0         
=================================================================
Total params: 1,154,485
Trainable params: 1,153,493
Non-trainable params: 992
```

- Hyperparameters values:

  Image size = 120*120
  Batch size = 50
  Number of frames (images) = 10
  Optimizer = Adam
  RGB Normalization value = 255
  Augmentation = False