DEEP LEARNING CASE STUDY - GESTURE RECOGNITION

By:
- Neeraj
- Praveen
- Thulasiram

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 recognise 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.

Dataset link: https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL

Plan for Case Study

1. **Visualize the data**
2. **Check for the class imbalance**
3. **Build Generator**
4. **Create a basic model to see if it trains well.**
5. **Start building models.**
6. **Validate the results using plots and its accuracy results.**
7. **Get the ways to develop the model to get better results.**
8. **Try using Different hyperparameters and create models using CONV3D, LSTM, GRU and even transfer learning, use augmentation even.**
9. **Repeat the step 5 to 8 until best results (atleast 85% accuracy is achieved with a smaller number of parameters.**
10. **Choose the best model and predict the class with the same.**

Observations from input data

The Training data consist of 663 videos having 30 frames each belonging to 5 classes as mentioned in the problem statement. These frames differ by the gestures that the people perform and the video is actually split and saved into image format (30 frames).

Carefully observing we could see most of the folders, the frames from 11 to 20 carry most of the important action belonging to the gesture, but then we can't eliminate the rest of the frames as such. But this can be an additional point that can help us while training the data

Class Imbalance check

From the class imbalance check on counting the number of folders belonging to each class we could see that there is a slight imbalance, but its not considerably high. 5^{th} class has only 123 training videos while others have 130+ data. And is validation dataset class 4 and 5 has only 16 and 17 respectively while others have 21+. So this can be neglected.

The Class imbalance check can be seen in the below image.

```
In [6]: # Checking for Class imbalance in training dataset

        pd.read_csv('Project_data/train.csv',sep=';')['0'].value_counts()
```

executed in 45ms, finished 07:54:07 2023-01-06

```
Out[6]: 1    137
        3    137
        0    135
        2    130
        4    123
        Name: 0, dtype: int64
```

```
In [7]: # Checking for Class imbalance in validation dataset

        pd.read_csv('Project_data/val.csv',sep=';')['0'].value_counts()
```

executed in 13ms, finished 07:54:07 2023-01-06

```
Out[7]: 1    23
        2    22
        3    21
        0    17
        4    16
        Name: 0, dtype: int64
```

Generator

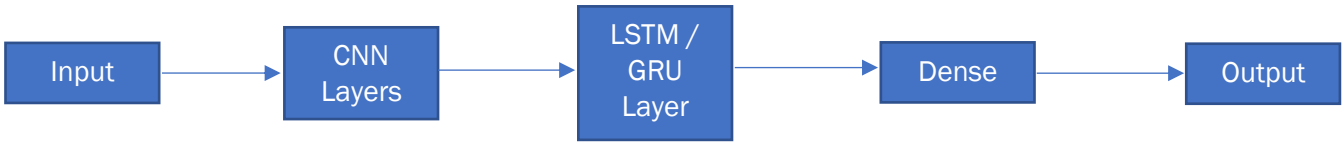
In the generator, we are going to pre-process the images as we have images of 2 different dimensions (360 x 360 and 120 x 160) as well as create a batch of video frames. The generator should be able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization are performed along with augmentation if it is preferred while training.

The Image size variation is also allowed so we can experiment with this also has a hyper parameter.

Architecture

We Used the following architectures with variations in the hyperparameters and depth of layers to get the best model performance.

- *3D Convolutional layer (CONV3D)*
- *RNN Layer - LSTM*
- *RNN Layer – GRU*
- *Transfer Learning with pretrained models.*



Architecture of RNN Model

Tabulation of the performance

Tabular Highlighting:

PURPLE - is the best performing model after considering the constraints of getting highest accuracy with lowest number of parameters.

GREEN - are also good performing models but with large number of parameters.

RED - OOM error (Memory outage / out of Memory).

Exp No.	Model Type	Batch Size	Result	Decision + Explanation	Parameters
1.1	CONV3D	64	Model Name - Base Model OOM Error Since GPU has limitations not able to train for 64 as batch size	Try with batch size 32.	2132469
1.2	CONV3D	32	Model Name - Base Model. Model Highly Over fits clearly Best Result - Epoch 10 Categorical Accuracy: 0.9985 Validation Accuracy: 0.3281	Reduce the parameters increase the layers and number of epochs	2132469
2.1	CONV3D	32	Model name - Model1 Model has overfitting again Best result - Epoch 13 Categorical Accuracy - 0.974702 Validation Accuracy - 0.242188	Try the same model with batch size 16	1940677
2.2	CONV3D	16	Model name - Model2 Model has trained good Best result - Epoch 23 Categorical Accuracy - 0.99256 Validation Accuracy - 0.892857	There is 10% difference between the training and validation Accuracy. We have used image size as 84*84, with 15 images per folder. We can try increasing the image size and try to add in some layers.	1940677
3.1	CONV3D	32	Model Name - Base Model OOM Error Since GPU has limitations not able to train for 32 as batch size due to model complexity.	Try training with batch size 16	7700933
3.2	CONV3D	16	Model name - Model2 Model has overfitted Best result - Epoch 19 Categorical Accuracy - 0.959821 Validation Accuracy - 0.830357	There is 10% difference between the training and validation Accuracy. We have used image size as 120*120, with 15 images per folder. We can try add in some more layers, but reduce the number of parameters.	7700933

4.1	CONV3D	16	Model Name - Base Model OOM Error Since GPU has limitations not able to train for 16 as batch size due to model complexity.	Try training with batch size 8	4577397
4.2	CONV3D	8	Model Name - Model3 Model has overfitted with low accuracy Best result - Epoch 9 Categorical Accuracy - 0.519578 Validation Accuracy - 0.307692	The model has many layers and model is trying to learn only the training data specific instead of being generic. We can try reducing the image size to 100*100 and also try increase the number of frames per folder to 19. We can try to reduce the number of parameters in next trial.	4577397
5	CONV3D	8	Model Name - Model4 Model has overcome the problem of overfitted with low accuracy Best result - Epoch 24 Categorical Accuracy - 0.739458 Validation Accuracy - 0.740385	Since reducing the parameters have good impact in reducing overfitting let us try to reduce it further. Initially let's try with 16 frames per video.	1507781
6	CONV3D	16	Model Name - Model5 Model has overfitted with low validation accuracy Best result - Epoch 24 Categorical Accuracy - 0.877976 Validation Accuracy - 0.705357	The model has many layers and model is trying to learn only the training data specific instead of being generic. We can try reducing the image size to 80*80 and also try with 16 frames. We can try to reduce the number of parameters in next trial. Also try with variations in learning rate reduction factor.	759605
7	CONV3D	16	Model Name - Model6 Model still has overfitting a little Best result - Epoch 14 Categorical Accuracy - 0.877976 Validation Accuracy - 0.705357	Try even reducing the parameters less than 200K. Include all frames this time and try.	391637
8	CONV3D	16	Model Name - Model7 Model started overfitting after 7th epoch Best result - Epoch 7 Categorical accuracy - 0.897321 Validation Accuracy - 0.803571	We could see the model has very well generalized but the accuracy is less. Let's try to make changes to model 6 with 120*120 image size and add few layers.	180345

9	CONV3D	16	Model Name - Model8 Model started overfitting after 7th epoch Best result - Epoch 15 Categorical accuracy - 0.965774 Validation Accuracy - 0.714286	Model is still overfitting. Let's try using RNN layers instead of CONV3D	759637
10	LSTM	15	Model Name - Model9 Model is performing good. Best result - Epoch 20 Categorical accuracy - 0.973333 Validation Accuracy - 0.885714	Let us increase the frames to 30 and reduce the LSTM nodes to 40 from 100.	787049
11		16	Model Name - Model10 Model is performing not that good. Kind of over fitting Best result - Epoch 25 Categorical accuracy - 0.979167 Validation Accuracy - 0.830357	Let us try the same model with GRU	717389
12	GRU	16	Model Name - Model11 Model is highly over fitting Best result - Epoch 17 Categorical accuracy - 0.997024 Validation Accuracy - 0.776786	Let us try to add in all the frames to LSTM model and try	710749
13	LSTM	16	Model Name - Model12Model is over fittingBest result - Epoch 17Categorical accuracy - 0.97619Validation Accuracy - 0.75	Change the patience value to 5 and again train Model12	717389
14		16	Model Name - Model13 Model is over fitting Best result - Epoch 21 Categorical accuracy - 0.937500 Validation Accuracy - 0.767857	Let us try augmentation techniques for the same model.	717389
Applying Data Augmentation					
15.1	LSTM	8	Model Name – Model14 Failed with OOM due to batch size 16 is heavy to process since model is little complex also image augmentation is included	Try with Batch size 8	717389
15.2	LSTM	8	Model Name - Model14 Model is well fitting Best result - Epoch 20 Categorical accuracy - 0.998485 Validation Accuracy - 0.905	Augmentation has definitely has helped improving the accuracy.	717389

16	CONV3D	16	Model Name - Model15 Model fits good until 12th epoch. Best result - Epoch 12 Categorical Accuracy - 0.928464 Validation Accuracy - 0.826923	Getting the Model6 from CNN and trying to use data augmentation using SGD optimizer. Let's try to use Adam with batch size as 12 and increase epochs to 30.	391637
17.1	CONV3D	16	Model Name - Model16 The Model training giving OOM because of 16 batch size.	Try with batch size 12	391637
17.2	Conv3D	12	Model Name - Model16 Model has given good results. Best result - Epoch 20 Categorical Accuracy - 0.922673 Validation Accuracy - 0.877451	Augmentation has definitely has helped improving the accuracy.	391637
18	LSTM	8	Model Name - Mode17 Model started to overfit after 15 epochs. Best result - Epoch 11 Categorical accuracy - 0.784849 Validation accuracy - 0.745	Getting the Model9 and applying augmentation and retraining with batch size 8.	787049
Using Pre trained Models					
19	Pretrained - Mobilenet	8	Model Name - Mode18 Model has fit well with pretrained model weights. Best result - Epoch 12 Accuracy - 0.987952 Validation Accuracy - 0.980769	Training without Augmentation.	3444485
20.1	Pretrained - Mobilenet	8	Model Name - Mode19 Model training resulted in OOM due to complex model	Try reducing batch size as 4	3513989
20.2	Pretrained - Mobilenet	4	Model Name - Mode19 The model has fit well with pretrained model weights. Best result - Epoch 12 Accuracy - 0.938822 Validation Accuracy - 0.935	Training with Augmentation.	3513989

21	Pretrained - VGG16	4	Model Name - Mode20 The model does not fit well with pretrained model weights. Best result - Epoch 22 Accuracy - 0.80136 Validation Accuracy - 0.72	Training with Augmentation.	14866693
22	Pretrained - InceptionV3	4	Model Name - Mode21 The model started overfitting after 10th epoch Best result - Epoch 6 Accuracy - 0.737915 Validation Accuracy - 0.71	Training with Augmentation.	22231031
23	Pretrained - MobileNetV2	4	Model Name - Mode22 The model gave very good validation accuracy and has fit well Best result - Epoch 23 Accuracy - 0.987915 Validation Accuracy - 0.965	Training with Augmentation.	2529559
24	Pretained - EfficientNetV2B0	4	Model Name - Mode23 The model gave very good validation accuracy and has fit well Best result - Epoch 15 Accuracy - 0.999245 Validation Accuracy - 0.975	Training with Augmentation.	6190887

Predictions

Prediction on Training data

Prediction on training data

In [14]:
```
train_path = r'Project_data\train'
train_doc = np.random.permutation(open('Project_data/train.csv').readlines())
test_batch = generator(source_path=train_path,folder_list=train_doc,ip_img_size=(80,80),
                       img_idx=[1, 3, 5, 7, 9, 11, 13, 15, 16, 17, 19, 21, 23, 25, 27, 29])
batch_data, batch_labels =next(test_batch)
```

```
Source path =  Project_data\train ; batch size = 16
```

In [15]:
```
# Training data
print(np.argmax(batch_labels,axis=1))
```

```
[3 3 0 2 0 3 4 0 2 0 2 1 1 2 3 2]
```

In [16]:
```
# Prediction
print(np.argmax(best_model.predict(batch_data[:,:,:,:,:]),axis=1))
```

```
1/1 [==============================] - 0s 65ms/step
[3 3 0 2 0 3 4 0 2 0 2 1 1 2 3 2]
```

Prediction on Validation data

Prediction on validation data

In [27]:
```
# Lets try with validation data
val_path = r'Project_data\val'
val_doc = np.random.permutation(open('Project_data/val.csv').readlines())
tval_batch = generator(source_path=val_path,folder_list=val_doc,ip_img_size=(80,80),
                       img_idx=[1, 3, 5, 7, 9, 11, 13, 15, 16, 17, 19, 21, 23, 25, 27, 29])
batch_data, batch_labels =next(tval_batch)
```

```
Source path =  Project_data\val ; batch size = 16
```

In [28]:
```
# validation data
print(np.argmax(batch_labels,axis=1))
```

```
[3 0 4 1 4 2 2 2 3 1 0 3 2 2 1 3]
```

In [29]:
```
# Prediction
print(np.argmax(best_model.predict(batch_data[:,:,:,:,:]),axis=1))
```

```
1/1 [==============================] - 0s 42ms/step
[3 0 4 1 4 2 2 2 3 1 0 3 2 0 1 3]
```

Summary of Observations and Conclusions:

- As number of trainable parameters increase, the model takes much more time for training.
- Batch size ∝ GPU memory / available compute. A large batch size can throw GPU Out of memory error, and hence we need to try with various batch sizes accordingly. Also, as the layers in the model increases memory usage will also increase and hence, we might need to reduce the batch size. The Models were trained on NVDIA RTX 3050 4GB GPU.

In [2]:
```
!nvidia-smi
```

```
Fri Jan  6 07:54:07 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 527.56       Driver Version: 527.56       CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| GPU  Name            TCC/WDDM | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ... WDDM  | 00000000:01:00.0 Off |                  N/A |
| N/A   53C    P0    13W /  75W |      0MiB /  4096MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

- Increasing the batch size greatly reduces the training time but can also have negative impact on the model accuracy. Thus, trade-off should be chosen wisely.
- When training with Augmented images we could see that overfitting was decreased and were able to achieve high accuracy models with least number of parameters possible.
- **Conv3D Model** was the best performing in terms of training accuracy and validation accuracy giving **92.2% and 87.7%** respectively also has least number of parameters < 4L (3,91,637 parameters).
- In LSTM Model the highest accuracy obtained was 99.8 and 90.5 for training and validation accuracy, but the number of parameters is nearly 2 times that of CONV3D model also the difference between the training and validation accuracy is high comparatively.
- Transfer learning boosted the overall accuracy for few models experimented. We made use of the MobileNet, VGG16, InceptionV3, MobileNetV2, EfficientNetV2B0.

- Of these, the highest performance was given by **EfficientNetV2B0** having least number of parameters among the pretrained models, giving **99.9% and 97.5%** training and validation accuracy respectively having parameters >60L (61,90,887).

- Even though the pretrained model gave a better performance the number of model parameters and the training time was high. In terms of efficiency, the model built using the Conv3D (Model 16) architecture gave the best results and hence we have shared the h5 file of that particular model.