DEEP LEARNING CASE STUDY **– *GESTURE RECOGNITION***

*Given Problem Statement*

As a Data Scientist working at a home electronics company (which manufactures state of art smart televisions). We need to develop a cool feature in the smart TV that can recognize five different gestures performed by the user which will help the users to control the TV without using a remote.

Following are the five gestures :

* *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 used :  [*https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL*](https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL)

*Dataset Properties :*

1. Training Data consists of a **few hundred videos** categorized into one of the **above five categories.**
2. **Each video** is typically **2-3 sec** in duration and divided into a sequence of total **30 frames**
3. Each subfolder within the train and val folder comprises of **one video each** (i.e total 30 frames/images).
4. All frames within a **subfolder are of equal size, but all subfolders vary in size** depending on the video they contain (mainly : 360\*360 or 120\*160). We have used functions *: min\_max\_scaler()* and *load\_image():* to scale images channel wise in order to maintain uniform scale per channel for all images.
5. CSV File has each **gesture mapped to a numeric label** between 0-4 for ease.

*To Do :* Train different models on Train data to predict the action performed in each video and test the model on Validation data. Finally choose the best model in terms of highest accuracy achieved on both training and validation set.

***Please Note :******All the steps followed have been clearly illustrated in the Notebook***

Architectures Used :

1. **3D Convolution Neural Network (Conv3D)** – *an extension of 2D-Convolution*

Here, we have a 3D kernel and filter moves in 3 directions – width, height and depth of an image (denoted by x, y and z here )

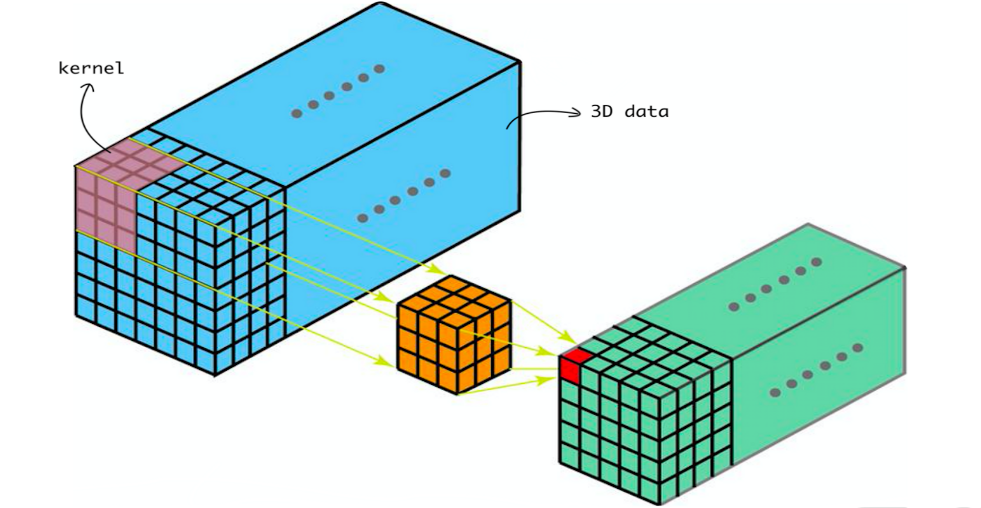


Fig – 3D CNN Model architecture

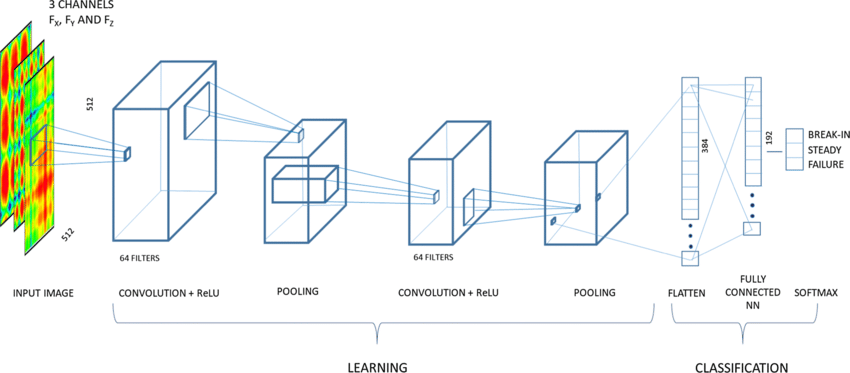
3D-max pooling

3D-convolution

3D-max pooling

FC-Dense Layer

3D-convolution

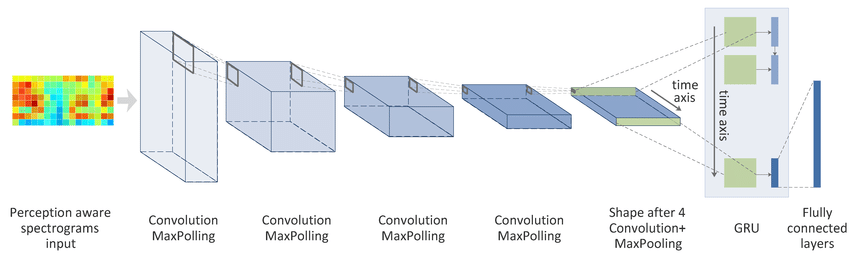


1. **CNN+RNN architecture**

CNN – conv2D network to extract a feature vector for each image

RNN – to apply on feature vector and give the output as predicted label (gesture)

Fig - CNN + GRU Model architecture



**MAJOR OBSERVATIONS**

1. Optimizers SGD() and Adam() were used for training models and it was observed that ***Adam was better*** in terms of **speed and accuracy**. The major advantage of using Adam was . However, the ***convergence rate was better with SGD (with a longer training time)***
2. **Dropouts and Data augmentation were used to rectify** overfitiing issues and it was observed that dropouts were most effective in preventing overfitting. However, it was also observed that **due to dropouts, *initial training accuracy remains slightly lower than validation. It improves with epochs and finally catches up with validation accuracy.***
3. ***ReduceLROnPleateau kept the Learning rate at check*** when val\_loss stopped improving over epochs. ***Early Stopping efficiently put a halt on training as the val\_loss started getting saturated.***
4. **Batch size heavily impacted the GPU memory** and a huge batch size threw Out of memory error in Google Colab. It was then reduced, which led to an increase in training time and brought down the accuracy. Finally, after further experimentation, an **optimal value for 16** was set which ***struck the right balance between computation time and accuracy (trade-off).***
5. **Transfer Learning boosted the overall performance** while keeping time complexity at check**. *MobileNet architecture outperformed VGGNet in terms of both complexity and accuracy.***
6. ***LSTM proved slightly better than GRU,*** *however in terms of complexity* ***GRU was far efficient, with less number of parameters. The final model had GRU layers along with MobileNet CNN.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MODEL | SR | RESULTS  (Accuracy and Loss) | DECISION + EXPLANATION | PARAMETERS Training Time |
| Conv3D (LR = 0.005)  Conv3D (LR = 0.002)  Conv3D (maxpooling added stride = 2,2,2)  Conv3D (added dropouts) – Model 1 in notebook  Conv3D (reduced parameters) | **1** | Out of Index Error | Incorrect number of batches specified. Rectified it | **-** |
| **2** | Training Accuracy : 0.95  Validation Accuracy : 0.81  Validation Loss : 1.15 | Overfitting and converging fast. Need early stopping and reduce Learning rate | **517,912**  *Total - 40 mins* |
| **3** | Training Accuracy : 0.80  Validation Accuracy : 0.74  Validation Loss : 1.21 | Val\_loss didn’t improve from much so early stopping stopped the training process.  *Need to add more layers & epochs* | **638,981**  *Total - 46 mins* |
| **4** | Training Accuracy : 0.91  Validation Accuracy : 0.81  Validation Loss : 0.67 | Accuracy improved but tending towards overfitting. Need to add dropouts. | **762,613**  *Total - 55 mins* |
| **5** | Training Accuracy : 0.93  Validation Accuracy : 0.84  Validation Loss : 0.44 | *Performance improved significantly after adding dropouts and increasing epochs. However, there is still a slight sense of overfitting.* | **485,637**  *95 mins -20 epochs* |
| **6** | Training Accuracy : 0.82  Validation Accuracy : 0.77  Validation Loss : 0.59 | Performance degraded | **356,533**  *Total - 25 mins* |
| CNN+RNN *(with GRU)*  MobileNet + GRU (*pretrained weights)*  *(3, 3, 3) Filter & 120 x 120 image resolution*  CNN+RNN *(with GRU)*  *(2,2,2) Filter and 84x84 img resolution*  CNN+RNN *(added a second GRU layer)*  VGGNet + GRU  *(LR = 0.002)*  VGG16 + *GRU (added another GRU layer)* | **CNN +RNN (with and without Transfer Learning)** | | | |
| **7** | Training Accuracy : 0.65  Validation Accuracy : 0.52  Validation Loss : 1.89 | Poor performance. Need to reduce filter size and image resolution. | **3,638,981**  *(12 mins per epoch)* |
| **8** | Training Accuracy : 0.75  Validation Accuracy : 0.72  Validation Loss : 1.21 | Accuracy improved but gets saturated with epochs. Need to add another GRU layer. | **1,762,613**  *(5 mins per epoch)* |
| **9** | Training Accuracy : 0.85  Validation Accuracy : 0.77  Validation Loss : 0.57 | Much better now. But number of parameters went up. Lets’ switch to Transfer Learning… | **2,556,533**  *(8-10 mins per epoch)* |
|  | **Transfer Learning** |  |  |
| **10** | Training Accuracy : 0.88  Validation Accuracy : 0.84  Validation Loss : 0.8 | Good performance in 10 epochs. Re-trained last few layers of VGGNet. Increased img resolution to 100x100, but no major difference in accuracy. | 11,013,045  *(42 mins per epoch)* |
| **11** | Training Accuracy : 0.90  Validation Accuracy : 0.85  Validation Loss : 0.46 | Significantly better performance in 10 epochs. But **time consuming** and increased parameters.  **(Model 2 in notebook)** | 14,859,045  *(50 mins per epoch)* |
| **12** | Training Accuracy : 0.90  Validation Accuracy : 0.63  Validation Loss : 0.77 | Overfitting is considerably high  *We are not training the MobileNet weights and can see, validation accuracy is very poor.* | 3,016,220  *(7 mins per epoch)* |
| MobileNet + GRU  *(re-training weights)* | **13** | Training Accuracy : 0.91  Validation Accuracy : 0.88  Validation Loss : 0.4 | Trained the MobileNet weights and kept a low learning rate (0.0008). **In just 6 epochs**, we achieved high training and validation accuracy. | 3,446,725  *(9-10 mins per epoch)* |
| MobileNet + GRU *reduced LR to 0.0004 (final model - 3 in nb)* | **14** | Training Accuracy : 0.93  Validation Accuracy : 0.92  Validation Loss : 0.30  *(least loss so far)* | *We re-trained some MobileNet weights and reduced LR to ~ 0.0004 witnessed significant improvement in accuracy in just 7 epochs.* | 3,446,725  *(10 mins per epoch)* |
| MobileNet +LSTM *( Replaced GRU with LSTM)* | **15** | Training Accuracy : 0.97  Validation Accuracy : 0.95  Validation Loss : 0.36 | Awesome results in 10 epochs !!  But increased parameters (*almost doubled*) | 7,692,869  (25 mins per epoch) |

FINAL MODEL

After multiple experiments with various parameters, **Model 14 - CNN\_GRU (MobileNet)** was finalized.

## **CNN+RNN (GRU) - using MobileNet (imagenet)**

### *Why MobileNet ?*

***MobileNet is 32 times smaller than VGG16, yet has the same accuracy*,** it is more efficient at capturing specific knowledge than VGG is.

It is also a highly efficient and lightweight convolutional neural network and not very computationally intensive either. *It resulted in a significantly high accuracy in about one-fifth of the time than VGG.*

**Final Model – Parameters Details**

***Base Model used****:*MobileNet***Weights used****:*imagenet

***Optimiser used****–*Adam***Learning rate****- 0.0004*

***Number of epochs****– 10* ***Batch-size****= 16*

***Image resolution used****= 100x100* ***Frames to sample****= reduced to 5*

***Activation used****-***RELU for Fully connected and Softmax for output layer**

Batch Normalization and Dropouts used in FC layers to keep a check on overfitting and vanishing/exploding gradients.

**Final Model – Metrics Details**

1. **Training Accuracy = 93% Validation Accuracy = 92%**
2. ***Training Loss* = 0.27 *Validation Loss* = 0.3**
3. **Total number of parameters** : 3,446,725

Chart

Description automatically generated