**Deep Learning Course Project - Gesture Recognition:**

*Authors: Adrish Maity, Sachin Malusare*

## Problem Statement:

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art **smart televisions**. You 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.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* 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

## **Understanding the Dataset**

The training data consists of a few hundred videos categorised into one of the five classes. Each video is divided into a **sequence of 30 frames(images)**. These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

Code divided into three major segments:

1. Generator:

2. Augmentation:

3. Model:

Initially simple base model with conv3D has been tried to check the correctness of the generator, augmentation and overall code flow. Some of the fundamental observations are confirmed and as expected. Same can be observed from the below table.

**a. Training time increases with increase in image size**

**b. Training time increases with number of sequences**

**Image Generator:**

As part of image generator, we have created a class where we have declared all the functions required to initialize the model and set the image parameters as well as the generator. We have setting variable image sizes, frames for different models as per model requirement.

There is a Boolean flag present in the generator function which controls the augmentation of the images which has a default value of False. In general images are feed into the model in a batch before that the images are cropped as per the height and width mentioned at the time of initialization followed by normalization of images.

**Image Augmentation Details:**

As part of augmentation there are 2 different kinds of augmentation have been applied to the images. The first one is the affine transform of the images based on the randomly chosen multiplier matrix followed by random rotation of 10 degree both sides.

The second augmentation technique applied here is the horizontal flipping of images. But the tricky thing here is that in case of horizontal flipping the left swipe and the right swipe are getting reversed. So, keeping that in mind we have updated the labels of the corresponding flipped images.

After applying the augmentation, the image dataset has increased which helps the model learn and generalize well.

**Experiment with different model architectures:**

We experimented with deeper network models with **conv3D** and **conv2d+RNN** **and Transfer Learning + RNN**.

**We have built the models keeping the size in mind.**

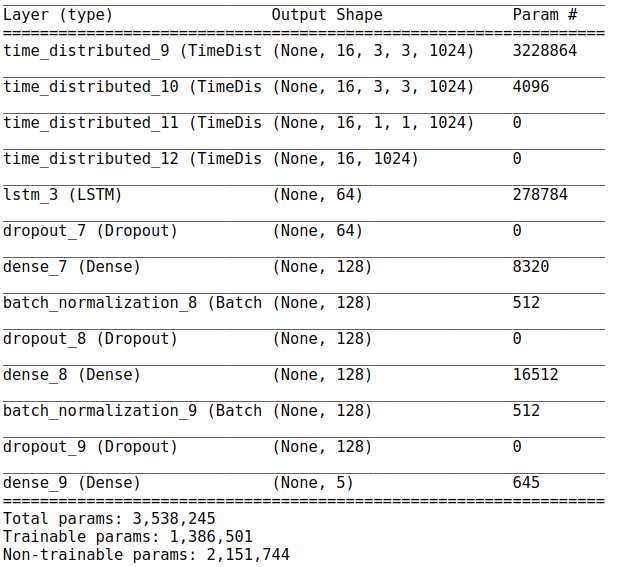
Observations and the explanations are tabulated as below:

**Results:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** Image: (160, 160) Frames: 18 Batch Size: 40 Trainable params: 1,116,325 Non-Trainable params: 736 Total params: 1,117,061 No. of Hidden layers: 3 No. of FC layers: 2 Epochs: 15 Learning Rate: 0.003  **No augmentation** | Best Result:  loss: 0.1695 categorical\_accuracy: **0.9744** val\_loss: 5.1562 val\_categorical\_accuracy: **0.28** | The model is overfitting. Need to reduce complexity. Also, we need to introduce data augmentation in this scenario. |
| **2** | **Conv3D** Image: (100, 100) Frames: 18 Batch Size: 10 Trainable params: 2,379,445 Non-Trainable params: 614 Total params: 2,380,069 No. of Hidden layers: 2 No. of FC layers: 1 Epochs: 20 Learning Rate: 0.0005 dropout in FC: 0.5  **Data augmentation has been applied** | Best Result:  loss: 0.35853 categorical\_accuracy: **0.87660** val\_loss: 0.66231 val\_categorical\_accuracy: **0.80**  Model file size: 28.6 MB | The model is not generalizing well in this configuration. We need to increase the number of layers as well as reduce the number of parameters to generalize the model. |
| **3** | **Conv3D** Image: (100, 100) Frames: 17 Batch Size: 10 Trainable params: 917,349 Non-Trainable params: 1,248 Total params: 918,597 No. of Hidden layers: 3 No. of FC layers: 2 Epochs: 20 Learning Rate: 0.0003 dropout in FC: 0.5  **Data augmentation has been applied** | Best Result:  loss: 0.52349 categorical\_accuracy: **0.80436** val\_loss: 0.4542 val\_categorical\_accuracy: **0.87**  Model file size: 11.1 MB | We have reduced the model input parameters and introduced dropout to regularize the model training. The validation accuracy is pretty decent. |
| **4** | **Conv3D** Image: (120, 120) Frames: 17 Batch Size: 10 Trainable params: 235,061 Non-Trainable params: 624 Total params: 235,685 No. of Hidden layers: 3 No. of FC layers: 2 Epochs: 25 Learning Rate: 0.0005 dropout in FC: 0.5  **Data augmentation has been applied**  For the last 2 Hidden layers kernel size has been reduced to (2,2,2) from (3,3,3) | Best Result:  loss: 0.28389 categorical\_accuracy: **0.90519** val\_loss: 0.57374 val\_categorical\_accuracy: **0.84**  Model file size: 2.9 MB | For this model we have reduced the parameters lesser than earlier model and achieved a good accuracy and excellent minimum model size of **2.9 MB** |
| **5** | **CONV2D + RNN (GRU)** Image: (120, 120) Frames: 18 Batch Size: 20 Trainable params: 2,680,005 Non-Trainable params: 2400 Total params: 2,682,405 No. of Hidden layers: 4 No. of FC layers: 2 Epochs: 25 Learning Rate: 0.0003 dropout in FC: 0.25 GRU units: 128  **Data augmentation has been applied** | Best Result:  loss: 0.3351 categorical\_accuracy: **0.8841** val\_loss: 0.6775 val\_categorical\_accuracy: **0.74**  Model file size: 31 MB | The model is overfitting. Need to reduce the complexity as well as the model size. |
| **6** | **CONV2D + RNN (GRU)** Image: (100, 100) Frames: 16 Batch Size: 10 Trainable params: 738,277 Non-Trainable params: 1,712 Total params: 739,989 No. of Hidden layers: 4 No. of FC layers: 2 Epochs: 20 Learning Rate: 0.0005 dropout in FC: 0.25 GRU units: 128  **Data augmentation has been applied** | Best Result:  loss: 0.43687 categorical\_accuracy: **0.84199** val\_loss: 0.64894 val\_categorical\_accuracy: **0.78**  Model file size: 9 MB | The model is performing decently and the model size is also reduced |
| **7** | **CONV2D + RNN (LSTM)** Image: (100, 100) Frames: 16 Batch Size: 10 Trainable params: 373,477 Non-Trainable params: 1,712 Total params: 375,189 No. of Hidden layers: 4 No. of FC layers: 2 Epochs: 20 Learning Rate: 0.0003 dropout in FC: 0.25 LSTM units: 32  **Data augmentation has been applied** | Best Result:  loss: 0.57054 categorical\_accuracy: **0.78631** val\_loss: 0.76256 val\_categorical\_accuracy: **0.72**  Model file size: 5 MB | The model is generalised very well. But accuracy is not that great. Let's try building the model using transfer learning |
| **8** | **Transfer Learning + RNN (GRU) using ImageNet** Image: (100, 100) Frames: 16 Batch Size: 20 Trainable params: 3,436,741 Non-Trainable params: 24,448 Total params: 3,461,189 No. of FC layers: 1 Epochs: 15 Learning Rate: 0.001 dropout in FC: 0.25 GRU units: 64  **Retrain all the weight in the ImageNet**  **Data augmentation has been applied** | Best Result:  loss: 1.17051 categorical\_accuracy: **0.96990** val\_loss: 1.17351 val\_categorical\_accuracy: **0.95**  Model file size: 41.7 MB | The model gives us a great result with 95% accuracy but the model size increases significantly. We need to reduce the model size keeping the accuracy in mind. |
| **9** | **Transfer Learning + RNN (GRU) using ImageNet** Image: (100, 100) Frames: 16 Batch Size: 20 Trainable params: 1,558,661 Non-Trainable params: 2,151,744 Total params: 3,710,405 No. of FC layers: 2 Epochs: 15 Learning Rate: 0.001 dropout in FC: 0.25 GRU units: 128  **Freeze initial 80 layers of ImageNet keeping the batch normalization trainable**  **Data augmentation has been applied** | Best Result:  loss: 3.82256 categorical\_accuracy: **0.86381** val\_loss: 3.76423 val\_categorical\_accuracy: **0.85**  Model file size: 27.6 MB | The model is generalised very well and both validation and test accuracy is very good. Let's try one more model with LSTM |
| **10** | **Transfer Learning + RNN (LSTM) using ImageNet** Image: (100, 100) Frames: 16 Batch Size: 20 Trainable params: 1,386,501 Non-Trainable params: 2,151,744 Total params: 3,538,245 No. of FC layers: 2 Epochs: 25 Learning Rate: 0.001 dropout in FC: 0.25 LSTM units: 64  **Freeze initial 80 layers of ImageNet keeping the batch normalization trainable**  **Data augmentation has been applied** | Best Result:  loss: 1.03546 categorical\_accuracy: **0.900068** val\_loss: 1.08622 val\_categorical\_accuracy: **0.89**  Model file size: **25.5 MB** | This model gives us very good accuracy and also the model size is optimum for webcam. We can choose this model for our final submission. |

We can see that **Transfer Learning (2D-Convolution) using MobileNet Architecture followed by RNN (LSTM)** gives us the better result in this case.

Final Model Architecture:



**Final Model: Validation Accuracy: 89%, Training Accuracy: 90%**