Gesture recognition

- Danish Ansari

Two models have been built

- **A CNN3D model**
- A CNN2D + RNN based model and

Summary of models is below:

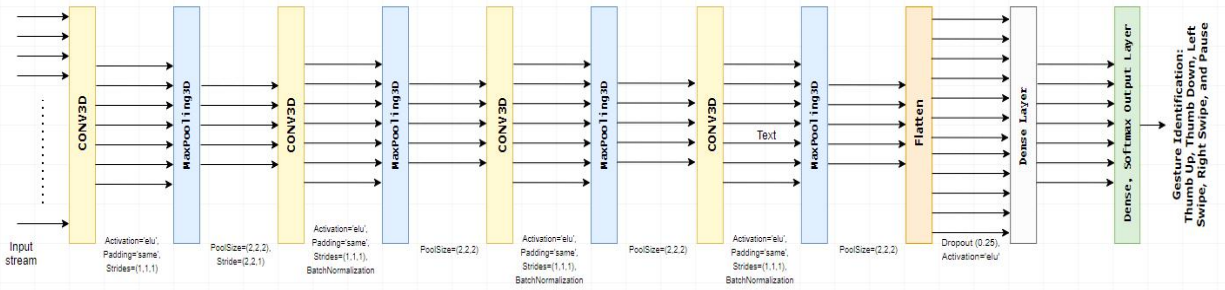
Conv3D Models

Experiment Number	Model	Result	Decision + Explanation
1	Conv3D – Tried to reach maximum batch size (This was tried on Colab and not Jarvis)	Broke at Batch size 50	Continued ahead with Batch size 40
2	Conv3D – Simple model with 2 Conv layers	Validation accuracy is not increasing.	Model is not learning at all. Maybe we have too less layers
3	Conv3D – Made a deeper network	Validation accuracy is not increasing.	Architecture change did not work at all. May be the batch size change can help
4	Conv3D – Previous model with reduced batch size of 25	Validation accuracy increased a bit, but train accuracy is constantly almost 100%	Model is still not learning. Should try making changes to architecture
5	Conv3D – Reduced conv layers to 2, also reduced the dense layer neurons. Applied only one BN as the first layer	There is some improvement in accuracy, but there seems to have been a sudden fall in loss in the first epoch for learning set, and	Model is still not learning. We might want to change the hyper-parameters and check

		then the model zeroed out	
6	Conv3D – Used more images per video, reduced the image size	Validation accuracy is still not increasing. Train accuracy is almost 100% from 4thepoch	Model is still not learning. Something must be wrong with the data generator.
7	Conv3D – Moved the notebook to Jarvis from this point. Nothing looks wrong with the generator. Chose elu activation instead of Relu, to avoid Dying Relu problem because that might be causing the dying out of loss.	Still validation accuracy is too low, and train accuracy is too high	Model is not learning. Last try with changing the architecture now
8	Conv3D – Made the model simpler while keeping the hyper-parameters same	There is some improvement with loss and accuracy	Improvement is still not enough. Thorough look is required into the generator
9	Conv3D – Went back and changed the normalization in the generator. Subtracted the mean per channel to center the values **[BEST MODEL IN TERMS OF ACCURACY]**	Train Accuracy = 86%, Val Accuracy = 81%	Model has leart. Problem was with previous normalization, where we had just divided by 255. But this model has too many parameters and will consume a lot of memory. Our business requirement is a light weight model. One trained model is consuming 28MB
10 **[FINAL MODEL]**	Conv3D – Made the model deeper but will less trainable parameters(only 234,549) **[BEST MODEL AS PER BUSINESS REQUIREMENT OF DECENT ACCURACY WITH LIMITED MEMORY SPACE]**	Train accuracy = 78% Val Accuracy = 74%	Performance is not as good as previous model, but the model consumes only 1MB as compared to the previous one that consumes 28MB. We should choose this one considering our business constraint that it must run in a smart TV

CONV3D Network

We performed few experiments by creating multiple networks of CNN2D + RNN (GRU). We found that the below model is simple having descent performance. The network of the model is as below:

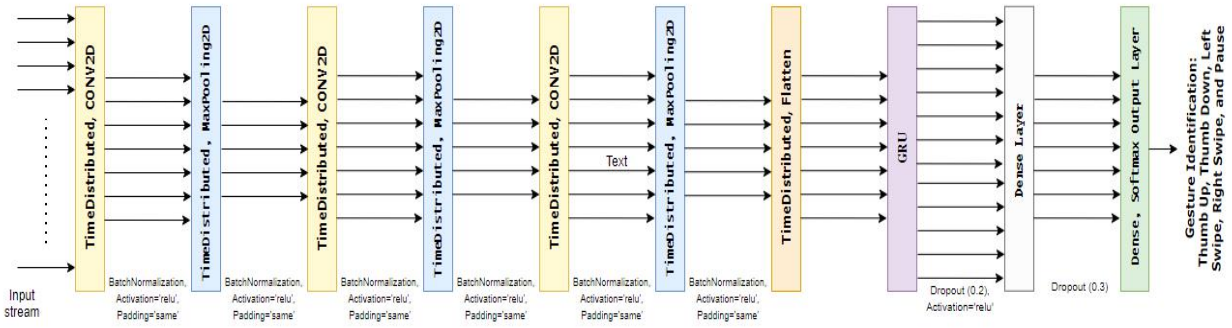


Conv2D + RNN

Experiment Number	Model	Result	Decision + Explanation
1	Conv2D+RNN (GRU)	Training categorical_accuracy: 84% Valuation categorical_accuracy: 66%	The accuracy is not great. But the model has learnt. Loss reduction has been constant for both train and validation. Number of parameters is high, so the model size will be big and is thus not suitable for our given business scenario
2	Conv2D+RNN (GRU) – Reduced parameters / Made deep	Training categorical_accuracy: 66% Valuation categorical_accuracy: 64%	This one is a considerably lighter model with less parameters.The performance is not too great but this is a better model than the previous one.
3	Conv2D+RNN (GRU) with Transfer Learning (MobileNet)	This model has not learned well, and it must not be used.	We used Transfer learning where we utilized MobileNet model, made all its layers starting from 55 as trainable, and froze the rest.

CNN + RNN(GRU) Network

We performed few experiments by creating multiple networks of CNN2D + RNN (GRU). We found that the below model is simple having descent performance. The network of the model is as below:



Approach

Imported the required libraries. The changes we made in the import section of sample code for reading and resizing images by using “imageio” and “skimage.transform” respectively.

Read metadata from .csv files

Load both train and validation .csv files in memory. Shuffle the records to break the order of the records so that the model learns from the training data.

Dataset of Classes, Videos and Gestures

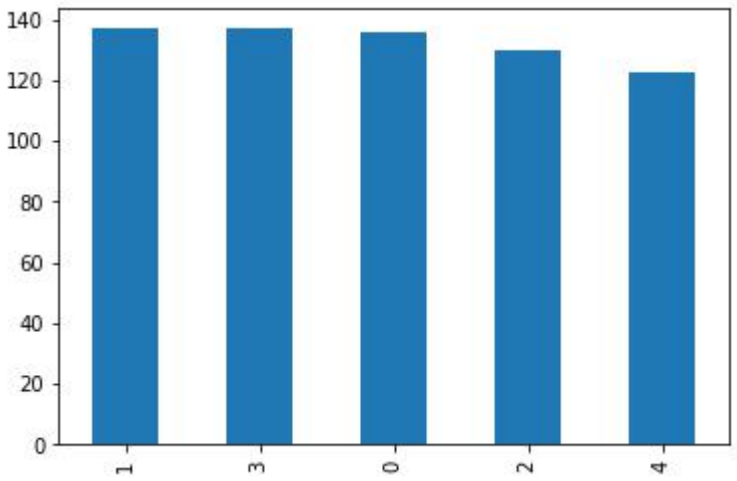
Create a data frame object with three columns: Video having gesture video clip path, Action for the gesture and the Class, the video clip belongs to.

Explore and understand the video frames

Explore the train dataset to understand it. There are 5 classes of gestures. These are

Class Name	Class Code
Left Swipe	0
Right Swipe	1
Pause	2
Thumbs Down	3
Thumbs Up	4

There is no class imbalance in the data. Plotting the dataset shows all the five classes are in proportion.



Load the image from the disk to find the dimensions of the video frames. The frames are of two different dimensions:

- (120, 160, 3) and
- (360, 360, 3)

We have resized all the images into the same shape before training.

Batch Generator function

We have parameterized the batch generation function with 3 extra parameters than what was provided in the sample code. Those are - number of frames, image height and image width. These help us to easily call the generator function with required parameters, when we want to change them, instead of having to hard code them.

The generator function scrambles the paths in the csv file for each epoch, so that we do not get class imbalance in the generated data, because the files are in order such that same class datapoints are together.

Generator produces data for each epoch in batches

Images are of two sizes in the dataset, so at the time of resizing, if it is the wider image with with 160, we will crop out the 20 from left and right leaving only 120 of the middle to resize. For the other size, we will normally resize as per the hyperparameter passed.

Image normalization has been done per channel for each image, by subtracting the mean of each channel for the entire train dataset from each channel of each image

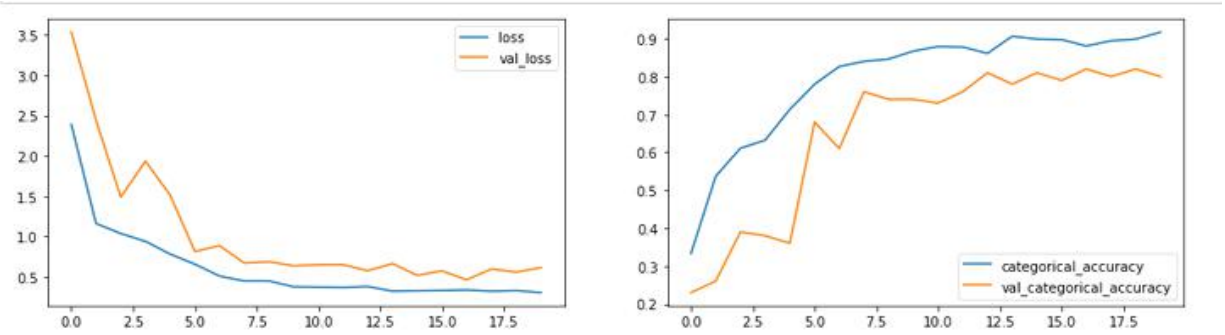
As per batch size, it is not necessary that the entire data can be divided perfectly into batches of the batch size. The last batch can contain the leftover datapoints (videos represented by number of images). Code has been separately written to handle that.

Plot the performance of the models

Plot for the 2 best Conv3D models are below:

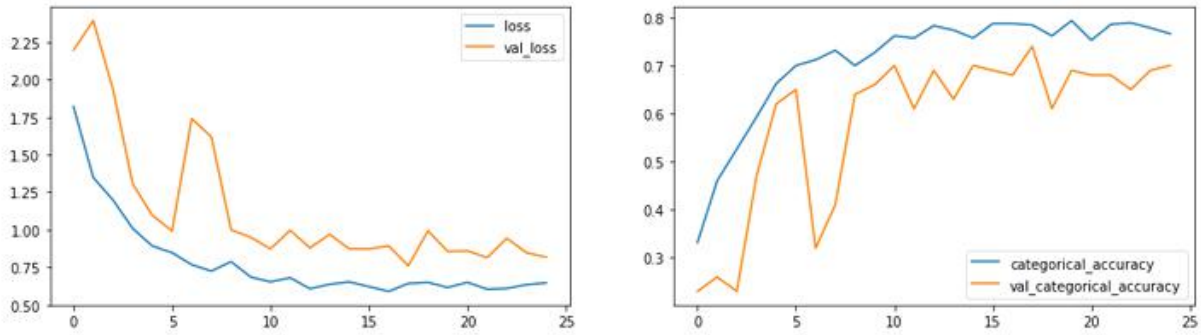
Conv3D Model Number 9 -

Train Accuracy = 86%, Val Accuracy = 81%, Number of trainable parameters: 7,008,293, Size on disk = 28MB



Conv3D Model Number 10 - [**FINAL MODEL**]

Train accuracy = 78%, Val Accuracy = 74%, Number of trainable parameters: 234,549, Size on disk = 1MB



Conv2D + RNN – Model 2:

Lightweight model, but not too good at accuracy

Training categorical_accuracy: 66% Valuation categorical_accuracy: 64%