# Gesture Recognition Project – Deep Learning

**Student:** Jheser Guzman

## Problem Statement

We need 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 following table consists of the experiments done to build a model to predict the gestures from the given data set.

| Model Iteration | Model | Hyper Parameters | Result | Explanation |
|---|---|---|---|---|
| 1 | Conv3D | Batch Size = 128<br>Ablation = 20<br>Augmentation = False<br>LR = 0.01<br>Seq Length = 10<br>Epoch = 20 | Train Accuracy: 0.15<br>Validation Accuracy: 0.15 | * The Model is not learning throughout the epochs.<br>* The loss is not decreasing.<br><br>I will reduce the batch size in the next experiment. |
| 2 | Conv3D | Batch Size = 32 | Train Accuracy: 0.14<br>Validation Accuracy: 0.21 | * Mdel does not improve<br><br>I will add more layers to the model in next experiment |
| 3 | Conv3D | | Train Accuracy: 0.21<br>Validation Accuracy: 0.19 | * Still there is no improvement in the model's accuracy.<br><br>I will add Batch normalization layers after every CNN and dense layers. |
| 4 | Conv3D | | Train Accuracy: 0.9053<br>Validation Accuracy: 0.2608 | * Model was able to over-fit on less data (Ablation data set),<br><br>I will train on full data and increasing epochs to 50. |
| 5 | Conv3D | Ablation = None<br>Epoch = 50 | Train Accuracy: 0.9172<br>Validation Accuracy: 0.71 | * Model is still over-fitting as there is a big difference between training and validation accuracies.<br><br>I will add some dropouts. |
| 6 | Conv3D | Drop out = 0.2 | Train Accuracy: 0.9686<br>Validation Accuracy: 0.7523 | * Slight Increase in the model validation accuracy and training accuracy.<br><br>I will increase the drop out values from 0.2 to 0.5 |
| 7 | Conv3D | Drop out = 0.5 | Train Accuracy: 0.9671<br>Validation Accuracy: 0.5023 | * Validation Accuracy dropped after the increase of the dropout (model is over-fitting).<br><br>I will use dropouts of 0.2 and I will remove the CNN layer to reduce the model complexity. |
| 8 | Conv3D | | Train Accuracy: 0.9412<br>Validation Accuracy: 0.8021 | * The model is performing well enough.<br>* Training and validation scores are good enough.<br>* The model has 710,533 trainable parameters. |
| 9 | Time Distributed + GRU | | Train Accuracy: 0.8680<br>Validation Accuracy: 0.8351 | * The model is performing better with less trainable parameters (98,885).<br><br>I will add more drop outs after each layer. |
| 10 | Time Distributed + GRU | Drop out = 0.2 | Train Accuracy: 0.8721<br>Validation Accuracy: 0.5611 | * Model accuracy decreased.<br><br>I will use GRU with a plain Dense Layer Network and some Global Avg Pooling. |
| 11 | Time Distributed + Dense | | **Train Accuracy: 0.8670**<br>**Validation Accuracy: 0.8450** | * Significant improvement in the Validation Accuracy<br>* Number of parameters 128,517.<br><br>I will try a different architecture of model with time distributed and ConvLSTM2D |
| 12 | Time Distributed + ConvLSTM2D | | **Train Accuracy: 0.9673**<br>**Validation Accuracy: 0.9375** | * Better results in Train and Validation Scores.<br>* number of parameters are 13,589<br><br>This is the best model so far!!! |

**Conclusion**

The best performing model was based on *Time Distributed Conv2D* and *ConvLSTM2D* (see attached Jupyter Notebook). It gave the best results so far compared to all the other models based only on Conv3D or TimeDistributed+GRU. The best performing model the least number of parameters as well compared with the other iterations.

**Future Work**

This experiment was performed using Anaconda3 on a M1 Max computer. The problem on this ARM64 architecture is the supported Metal libraries for TensorFlow to use the GPU. Using the GPU would speed up all the experimentation in many orders of magnitude, since it was using CPU only for now.