Gesture Recognition

# Problem statement

# Recognising 5 different hand gestures to control a smart TV

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 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' backward 10 seconds

- Right swipe: 'Jump' forward 10 seconds

- Stop: Pause the movie

The training data consists of a few hundred videos categorized into one of the five classes. Each video (typically 2-3 seconds long) 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.

| **Experiment Number** | **Model** | **Hyper-Parameters** | **Result** | **Decision + Explanation** |
| --- | --- | --- | --- | --- |
| **1** | **Conv 3D** | **Batch size = 128, Ablation = 20, Augmentation = False, LR = 0.01, Seq Length = 10, Epoch = 20, Dim = 120x120** | **Train Accuracy: 0.15, Validation Accuracy: 0.15** | **The Model is not learning anything throughout the epochs, the loss is not decreasing. Reducing the batch size further.Trainable parameters -12,904,581** |
| **2** | **Conv 3D** | **Batch size = 32** | **Train Accuracy: 0.15, Validation Accuracy: 0.20** | **No improvement in the model, lets add more layers to the model so that it can learn from data. Trainable parameters -12,904,581** |
| **3** | **Conv 3D** |  | **Negative Dimension Error** | **The new CNN kernel sizes are not compatible with the output of previous layers. Let’s reduce the kernel size of new layers.** |
| **4** | **Conv 3D** |  | **Train Accuracy: 0.20, Validation Accuracy: 0.20** | **Still there is no improvement in the model. Let’s add Batch normalization layers after every CNN and dense layers.Trainable parameters - 8,311,813** |
| **5** | **Conv 3D** |  | **Train Accuracy: 0.9623, Validation Accuracy: 0.2812** | **Model is able to over-fit on less data (Ablation data set), Let’s Training on full data and increasing epochs to 50. Trainable parameters - 8,314,757** |
| **6** | **Conv 3D** | **Ablation = None, Epoch = 20** | **Train Accuracy: 0.9623, Validation Accuracy: 0.15** | **Model is over-fitting as there is huge gap between training and validation accuracies. Let’s add some dropouts so that the model can be generalized. Trainable parameters - 8,314,757** |
| **7** | **Conv 3D** | **Dropout = 0.2** | **Train Accuracy: 0.9905, Validation Accuracy: 0.3594** | **Not much improvement in the results so we can try to increase the dropout from 0.2 to 0.5. Trainable parameters - 8,314,757** |
| **8** | **Conv 3D** | **Dropout = 0.5** | **Train Accuracy: 0.9657, Validation Accuracy: 0.172** | **After increasing the dropout, the model validation score further reduced and the model was over-fitted. Let’s use 0.2 as dropout and we will remove a CNN layer to reduce the complexity of the model. Trainable parameters - 8,314,757** |
| **9** | **Conv 3D** | **Dropout = 0.2** | **Train Accuracy: 0.99, Validation Accuracy: 0.125** | **Still the model is over-fitting and the validation accuracy results are not good. Let’s use a Global Average Pooling instead of Flatten Layer. Trainable parameters - 22,730,629** |
| **10** | **Conv 3D** |  | **Train Accuracy: 0.965, Validation Accuracy: 0.125** | **The model does not showcase any improvement. We can try different architectures. Trainable parameters - 710,533** |
| **11** | **Time distributed + GRU** |  | **Train Accuracy: 0.801, Validation Accuracy: 0. 273** | **The model is working lower than expectations on the validation dataset with less trainable parameters(99,269), Lets add some drop outs after each layer, so that both train and validation accuracies will be closer and improve. Trainable parameters - 99,269** |
| **12** | **Time distributed + GRU** | **Dropout = 0.2** | **Train Accuracy: 0.735, Validation Accuracy: 0. 172** | **The model accuracy further deteriorated; Let’s replace GRU with a plain Dense Layer Network and some Global Avg Pooling. Trainable parameters - 99,269** |
| **13** | **Time distributed + Dense** |  | **Train Accuracy: 0.852, Validation Accuracy: 0. 125** | **The model accuracy further deteriorated; Let’s replace GRU with a plain Dense Layer Network and some Global Avg Pooling. Trainable parameters - 128,517** |
| **14** | **Time Distributed + ConvLSTM 2D** |  | **Train Accuracy: 0.724, Validation Accuracy: 0. 36** | **The model accuracy further deteriorated; Let’s train using some other method i.e. CNN+LSTM. Trainable parameters - 13,589** |
| **15** | **CNN\_LSTM** |  | **Train Accuracy: 0.78**  **Val Accuracy:**  **0.79**  **Total parameters: 1 005 541** | **We use the new generator code from experiment 15 Train a model base with CNN + LSTM. The accuracy will be better with more epochs.** |
| **16** | **CNN\_LSTM with Transfer learning** |  | **Train Accuracy: 0.996**  **Val Accuracy:**  **0.74**  **Total parameters: 3 228 864** | **From experiment 15, instead of training the CNN\_LSTM with more epochs, we decide to train by transfer learning. The training accuracy is very high but the valuation accuracy is not. So It can be overfitting.** |
| **Final Model** | **ConvLSTM2D** |  | **Train Accuracy: 0.985**  **Val Accuracy:**  **0.9**  **Total parameters: 13 781** | **From all previous experiments, we decide to use convLSTM2D to learn more and handle the overfitting problem. This helps us have good results** |