Gesture Recognition – Deep learning

# Problem Statement:

The advancement of smart television technology has introduced new opportunities for user interaction and control. However, traditional remote controls can sometimes be cumbersome or prone to loss. To enhance user experience and convenience, our objective is to develop a gesture recognition feature for smart televisions. This feature will enable users to control their television sets through hand gestures captured by the built-in webcam.

The challenge at hand is to design and implement a robust gesture recognition system capable of accurately identifying five distinct hand gestures: thumbs up, thumbs down, left swipe, right swipe, and stop. Each gesture corresponds to a specific command, such as increasing or decreasing the volume, navigating backward or forward in a video, and pausing playback.

To achieve this, we have access to a dataset comprising several hundred videos, each containing sequences of 30 frames captured by a webcam. These videos are categorized into one of the five classes based on the gesture performed by the user. Our task is to develop machine learning models or deep learning architectures capable of effectively classifying these gestures in real-time, thus providing users with an intuitive and seamless alternative to traditional remote controls.  
  
The table below outlines the experiments conducted in order to construct a model for predicting gestures from the provided dataset.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Exp. #** | **Model** | | **Hyper- parameters** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | | **Batch size = 20,**  **LR = 0.01,**  **Epoch = 15,**  **Dim = 120x120** | **Train Accuracy: 1.00,**  **Train Loss : 0.0053**  **Validation Accuracy: 0.77**  **Validation Loss : 1.08** | **To enhance the model's performance, let's augment it with additional layers to enable more comprehensive learning from the data.** |
| **2** | **Conv3D** | | **Batch size = 20,**  **LR = 0.01,**  **Epoch = 15,**  **Dim = 120x120** | **Train Accuracy:0.2045,**  **Train Loss : 1.59**  **Validation Accuracy: 0.220**  **Validation Loss : 1.61** | **The model shows stagnation in learning across epochs as the loss fails to decrease.** |
| **3** | **Conv3D** | | **Batch size = 10,**  **LR = 0.001,**  **Epoch = 20,**  **Dim = 120x120**  **Frames =20** | **Train Accuracy:0.9866,**  **Train Loss : 0.05**  **Validation Accuracy: 0.830**  **Validation Loss : 0.568** | **To enhance the model's performance, let's augment it with additional layers to enable more comprehensive learning from the data.** |
| **4** | **Conv3D** | | **Batch size = 10,**  **LR = 0.002,**  **Epoch = 30,**  **Dim = 120x120**  **Frames =20** | **Train Accuracy:0.8821,**  **Train Loss : 0.32**  **Validation Accuracy: 0.900**  **Validation Loss : 0.323** | **This model exhibits commendable performance, boasting impressive training and validation accuracies, with a configuration consisting of 710,533 trainable parameters and 1,920 non-trainable parameters. Let's explore a different architectural approach to further optimize its capabilities.** |
| **5** | **Conv3D** | | **Batch size = 10,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120,**  **Dropout = 0.25** | **Train Accuracy:0.925,**  **Train Loss : 0.2226**  **Validation Accuracy: 0.660**  **Validation Loss : 1.031** | **The model is currently experiencing overfitting, evident from the substantial disparity between training and validation accuracies. Let's integrate dropout layers to enhance its ability to generalize.** |
| **6** | **Conv3D** | | **Batch size = 20,**  **LR = 0.002,**  **Epoch = 30,**  **Dim = 120x120,**  **Dropout = 0.25** | **Train Accuracy:0.500,**  **Train Loss : 1.044**  **Validation Accuracy: 0.610**  **Validation Loss : 0.98** | **The model's performance suggests some level of improvement is needed to achieve better generalization.** |
| **7** | | **Time Distributed**  **+ GRU** | **Batch size = 15,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120** | **Train Accuracy:0.8652,**  **Train Loss : 0. 3651**  **Validation Accuracy: 0.6952**  **Validation Loss : 0.68** | **The model is performing admirably on the validation dataset despite having fewer trainable parameters. To further balance the train and validation accuracies, let's incorporate dropout layers after each layer.** |
| **8** | | **Time Distributed**  **+ GRU** | **Batch size = 15,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120,**  **Frames =20** | **Train Accuracy:0.808,**  **Train Loss : 0.446**  **Validation Accuracy: 0.600**  **Validation Loss : 1.07** | **The model is performing admirably on the validation dataset despite having fewer trainable parameters.** |
| **9** | | **Time Distributed**  **+ GRU** | **Batch size = 15,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120,**  **Frames =20** | **Train Accuracy:0.800,**  **Train Loss : 0.463**  **Validation Accuracy: 0.714**  **Validation Loss : 0.598** | **The model's accuracy has declined even further. To address this, let's substitute the GRU layer with a straightforward Dense Layer Network and introduce some Global Average Pooling.** |
| **10** | | **Time Distributed**  **+ Dense** | **Batch size = 25,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120,**  **Frames =20** | **Train Accuracy:0.8593,**  **Train Loss : 0.3901**  **Validation Accuracy: 0.500**  **Validation Loss : 1.3042** | **The current model demonstrates satisfactory training and validation accuracies, utilizing a total of 128,517 parameters. Let's explore an alternative architecture incorporating time distributed layers and ConvLSTM2D.** |
| **11** | | **Time Distributed**  **+**  **ConvLSTM 2D** | **Batch size = 15,**  **LR = 0.002,**  **Epoch = 20,**  **Dim = 120x120,** | **Train Accuracy:0.8074,**  **Train Loss : 0.5057**  **Validation Accuracy: 0.7810**  **Validation Loss : 0.5082** | **We've achieved the optimal model performance thus far. The validation accuracy is impressive, and with only 13,589 parameters, the model boasts a compact size of just 226KB.** |

# Conclusion:

Experiment 11 and Experiment 4, featuring a model constructed with Time Distributed Conv2D and ConvLSTM2D layers, yielded superior results compared to all previous models. Additionally, it boasts the fewest parameters among them.