# **Gesture Recognition Model**

This project implements a gesture recognition system using a deep learning pipeline. The model leverages sequential data from body landmarks extracted using Mediapipe and employs Bidirectional GRU (Gated Recurrent Units) for robust temporal pattern recognition. Below is an overview of the project and its components.

## **Key Features**

* **Input Data**: 45 frames per gesture, with 258 features per frame (landmarks from pose and hands only).
* **Model Architecture**:
  + Two Dense (DNN) layers for feature extraction.
  + Three Bidirectional GRU layers for temporal modeling.
  + Batch Normalization layers for improved training stability.
  + Final Dense layers for classification.
* **Output**: 6-8 gesture classes, depending on the dataset configuration.
* **Visualization**: Real-time prediction with probability visualization for gesture confidence.

## **Dataset Preparation**

1. **Data Source**:  
   * Extracted Mediapipe landmarks: Pose (33 points) + Hands (21 points each).
   * Face landmarks were excluded to improve model clarity and reduce confusion.
2. **Preprocessing**:  
   * Each gesture sequence is converted into a NumPy array with dimensions (45, 258).
   * Data normalization ensures consistent input for the model.
   * Principal Component Analysis (PCA) is used to reduce input feature dimensions if needed.

## **Model Overview**

The final model was designed to balance complexity and performance, achieving optimal accuracy on limited hardware:

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense, GRU, BatchNormalization, Bidirectional

model = Sequential()

# DNN Layers

model.add(Dense(256, activation='relu', input\_shape=(45, 258)))

model.add(Dense(128, activation='relu'))

# Bidirectional GRU Layers

model.add(Bidirectional(GRU(256, return\_sequences=True)))

model.add(Bidirectional(GRU(128, return\_sequences=True)))

model.add(Bidirectional(GRU(64)))

# Dense Layers

model.add(Dense(64, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(8, activation='softmax')) # For 8 gesture classes

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

model.summary()

## 

## **Training Details**

* **Batch Size[4]**: Optimized to balance memory and convergence rate.
* **Learning Rate**: 1e-4 using Adam optimizer for stable training.
* **Epochs**: 100 epochs to allow sufficient learning without overfitting.

## **Real-Time Prediction**

1. **Pipeline**:  
   * Captures real-time video using OpenCV.
   * Extracts keypoints from each frame using Mediapipe.
   * Predicts gestures based on the most recent 45 frames.
2. **Visualization**:  
   * Probabilities for all classes are visualized as colored bars.
   * Top prediction is displayed prominently on the video feed.

## **Challenges and Solutions**

* **Face Landmarks Overload**:  
  + Face landmarks were initially included, contributing ~1400 features out of 1600. This caused model confusion due to their static nature.
  + Solution: Face landmarks were excluded, focusing on pose and hand dynamics.
* **Hardware Limitations**:  
  + Model size and data preprocessing were optimized to ensure reasonable training and inference times.
* **Data Quality**:  
  + Early datasets were suboptimal. Higher-quality data significantly improved the model's accuracy and ability to generalize.

## **Performance**

* **Training Accuracy**: Satisfactory accuracy achieved on the refined dataset.
* **Inference Time**: ~19ms/step, suitable for real-time applications.

## **Future Work**

* **Augmentation**: Adding data augmentation techniques for robustness.
* **Model Optimization**: Quantization and pruning for faster inference on edge devices.
* **Additional Classes**: Expanding gesture classes for broader use cases.

## **Requirements**

* Python 3.7+
* TensorFlow 2.x
* Mediapipe
* OpenCV
* NumPy