1. Baseline Setup & Motivation

1.1 Dataset Overview

- Each gesture video is 30 frames, classified into one of 5 categories (Thumbs Up, Thumbs Down, Left Swipe, Right Swipe, Stop).
- Goal: Build a model that classifies these gestures accurately while balancing model size (for real-time inference on smart TVs).

1.2 Initial Architecture

- **CNN + RNN** approach:
    - **TimeDistributed** Convolutional layers to extract spatial features from each frame.
    - A **GRU** layer to capture temporal sequences across frames.
    - Final **Dense** layers for classification into 5 classes.

1.3 Early Results

- We started with two conv layers (`16, 32` filters), stride or pooling for downsampling, and `GRU(32)`.
- **Initial metrics** were ~60–70% training accuracy and ~45–50% validation accuracy, indicating underfitting and some overfitting signs.

---

2. Preprocessing & Data Handling

2.1 Frame Selection

- Instead of using all 30 frames, we typically **sampled 15 frames** (every other frame) to reduce computational load.

2.2 Cropping & Resizing

- **Experiment**: Cropping out unnecessary borders or background. Moderate cropping helped if gestures were centered, but we had to avoid losing key motion on the edges.
- **Resize**:
    - Switched to **64×64** frames (down from 100×100) to reduce GPU memory usage and speed up training.
    - Adjusted pooling steps accordingly (two `MaxPooling2D(2×2)` layers => final flattened dimension ~8192).

2.3 Normalization

- Simple division by 255.0 to get pixel values in [0,1]. This kept the CNN stable during training.

2.4 Custom Generator

- Reads subfolders for each video, loads frames, applies preprocessing, then yields `(batch_data, batch_labels)`.
- Shuffles data each epoch and can handle leftover samples.

---

3. Model Evolution & Experiments

3.1 Adding a Third Convolutional Layer

- **Reason**: Deeper CNN might capture richer spatial features.
- **Observation**: Training accuracy improved by ~5%, validation by ~3%. Some risk of additional overfitting if not combined with dropout.

3.2 Filter Sizes & GRU Units

- Tried `(8,16)`, `(16,32)`, `(16,32,64)` for conv filters. Larger filters improved training accuracy but needed more regularization.
- **GRU**: We tested 16 vs. 32 units. **32** gave ~2–4% better validation accuracy while doubling the parameter count.

3.3 Regularization

- **Dropout**: Inserted after `Flatten`, inside the GRU (`dropout=0.2, recurrent_dropout=0.2`), and sometimes after a Dense(64) layer. Helped reduce overfitting by ~5–7% in the training-validation accuracy gap.
- **EarlyStopping & ReduceLROnPlateau**:
    - EarlyStopping (patience=10) prevented unnecessary epochs once the model began to overfit.
    - ReduceLROnPlateau (patience=5, factor=0.1) helped find a smaller learning rate if the validation loss stagnated.

3.4 Data Augmentation

- Explored simple horizontal flips, slight random cropping, and brightness shifts. Some gestures (e.g., Left vs. Right Swipe) might be sensitive to flips, so augmentation needed to be carefully tuned.

4. Final Results & Observations

1. **Training Accuracy**: **~97.43%**
   - The model is almost perfect on the training set, with a low training loss of **0.1162**.
2. **Validation Accuracy**: **~80%**
   - The validation accuracy is good but shows a gap of ~17% compared to the training set.
   - **Validation Loss**: **0.7871**—significantly higher than the training loss, reflecting **overfitting**.
3. **Learning Rate**: **0.001**
   - A standard starting rate for Adam. With such a high training accuracy, the limiting factor is generalization rather than convergence speed.

Interpretation:

- The large gap between training (~~97.43%) and validation (~~80%) indicates the model is **overfitting**. Even though we introduced dropout and used EarlyStopping, the network has learned the training set extremely well and doesn't generalize as strongly as we'd like.

Potential Next Steps

1. **Increase Regularization**:
   - Add more dropout, or raise dropout rates (e.g., 0.3).
   - Explore L2 weight decay on convolution and GRU layers.
2. **Refine Data Augmentation**:
   - More varied transforms (random zoom, shifts, or partial rotations) to boost diversity.
3. **Reduce Model Complexity**:
   - If feasible, decrease the number of Conv filters or reduce GRU units from 32 to 16. This might lower training accuracy but could also close the gap by preventing memorization.
4. **Transfer Learning**:
   - Use a pretrained CNN (e.g., MobileNet) in TimeDistributed fashion and only fine-tune top layers.

5. Conclusion

Through systematic adjustments—**frame resizing**, **layer additions**, **dropout**, and **data augmentation**—we achieved:

- **~97.43%** on training data
- **~80%** on validation data

The model now clearly demonstrates **high capacity** but also **overfitting**. Future efforts will focus on **improving generalization** via stronger regularization or transfer learning while maintaining real-time performance on embedded hardware.