Hand Gesture Recognition using Deep Learning

1. Introduction

This project develops a deep learning system to recognize dynamic hand gestures from video sequences, facilitating touchless human-computer interaction. The system uses Transfer Learning with pre-trained CNNs and a Time-Distributed architecture to handle temporal video data.

2. Dataset & Preprocessing

- **Source:** Kaggle Hand Gesture Recognition Dataset.
- **Classes:** Left_Swipe, Right_Swipe, Stop, Thumbs_Down, Thumbs_Up.
- **Preprocessing:**
    - Frame Extraction: Videos decomposed into frames.(30)
    - Resizing: Standardized to 64x64 pixels.
    - Normalization: Pixels scaled to [0, 1].
    - Augmentation: Horizontal flipping applied to balance classes.
    - Serialization: Saved as .npy files with metadata management.

3. Methodology

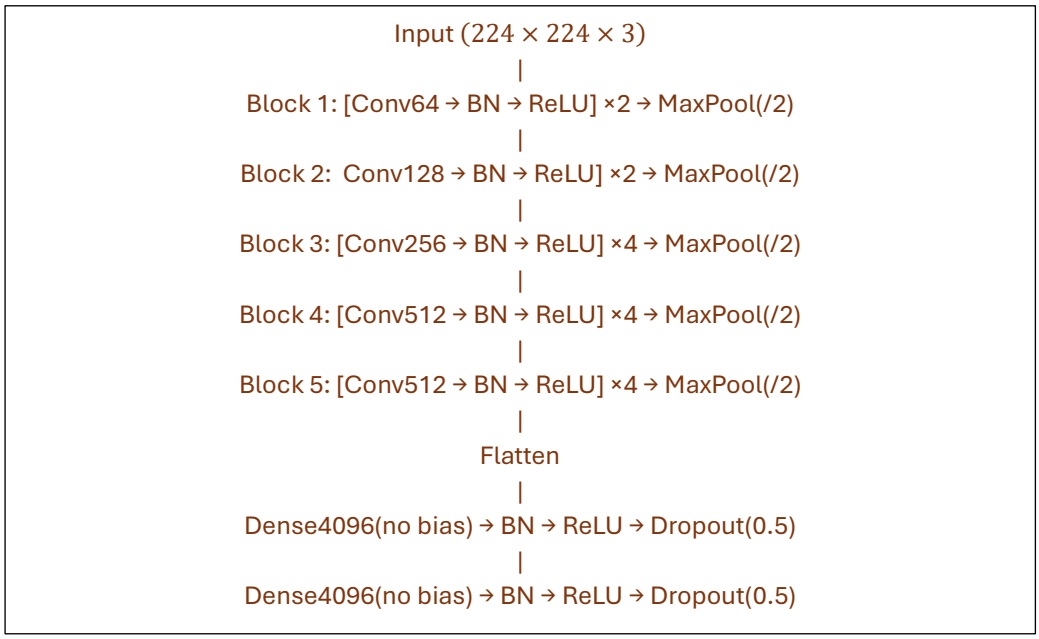
Three models were implemented using Transfer Learning. All follow a consistent architecture designed for video processing.

All models process video batches of shape (30 Frames, 64, 64, 3)

1. Model 1:VGG19

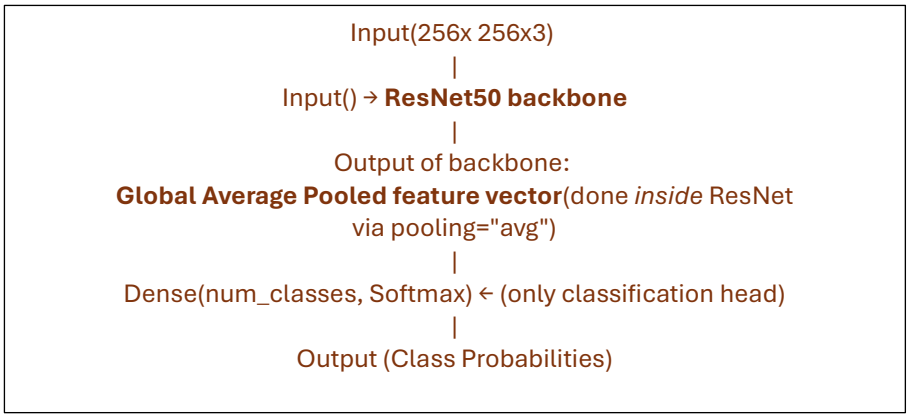
- **Base:** Custom VGG19-style CNN (built with Conv2D + BatchNorm + ReLU blocks).
- **Focus:** Classic deep CNN feature learning (stacked 3×3 convolutions + max pooling).
- **Training:** Trained from scratch (all layers trainable; no pretrained backbone).
- **Overall structure**
    - Input: (224×224×3)
    - Backbone (convolutional feature extractor):
    - Block 1: [Conv3×3(64) → BN → ReLU] ×2 → MaxPool(/2)
    - Block 2: [Conv3×3(128) → BN → ReLU] ×2 → MaxPool(/2)
    - Block 3: [Conv3×3(256) → BN → ReLU] ×4 → MaxPool(/2)
    - Block 4: [Conv3×3(512) → BN → ReLU] ×4 → MaxPool(/2)
    - Block 5: [Conv3×3(512) → BN → ReLU] ×4 → MaxPool(/2)
- **Top / head:**
    - Flatten
    - Dense(4096, use_bias=False) → BN → ReLU → Dropout(0.5)
    - Dense(4096, use_bias=False) → BN → ReLU → Dropout(0.5)
    - Dense(num_classes=5, activation="softmax")

- **Trainability pattern**
  - Phase 1: all layers trainable (trained end-to-end from scratch)
  - Phase 2: none (no freezing / fine-tuning stage in the notebook)
- **Diagram:**



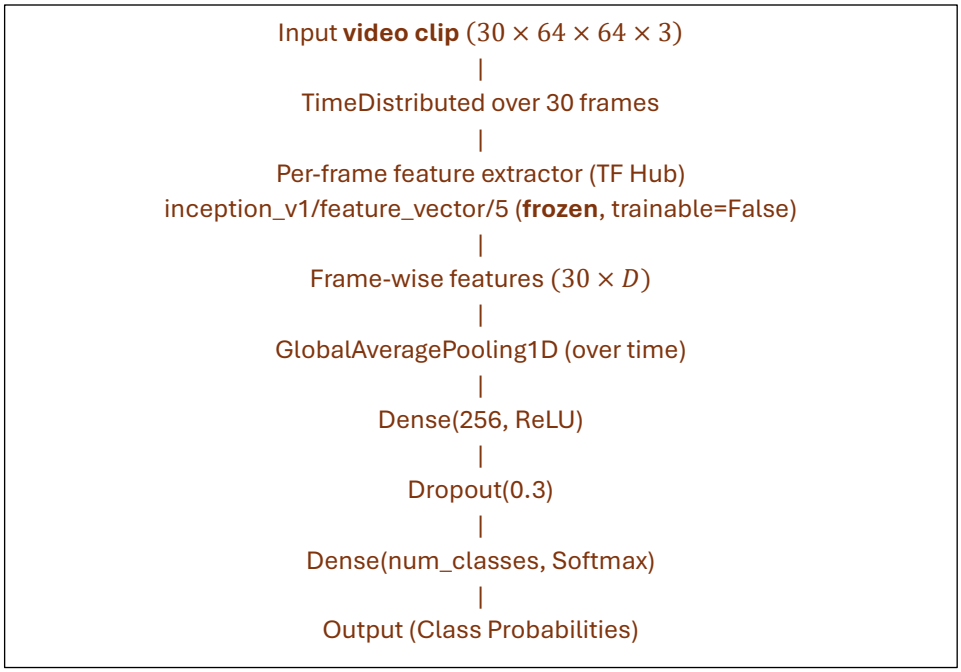
2. Model 2: ResNet50

- **Base:** ResNet50.
- **Focus:** Deep network training via residual connections.
- **Training:** Frozen base (Feature Extraction only).
- **Overall structure**
  - Input: (256, 256, 3)
  - Backbone: ResNet50(include_top=False, weights="imagenet", pooling="avg")
- **Top / head:**
  - Dense(num_classes, activation="softmax")
  - Output: class probabilities
- **Trainability pattern**
  - Phase 1: backbone frozen (base.trainable = False)
  - Phase 2: backbone partially fine-tuned (only last UNFREEZE_LAST_LAYERS layers trainable; earlier layers frozen)
- **Diagram:**



3. Model 3: InceptionV1

- **Base:** InceptionV1 (TensorFlow Hub).
- **Focus:** Multi-scale feature capture using Inception modules.
- **Training:** Frozen base (Feature Extraction only).
- **Overall structure**
  - Input: $(30, 64, 64, 3)$ *(30-frame clip)*
  - Per-frame backbone (frozen): TF-Hub
  - TimeDistributed (Lambda(feature_extractor)) over the 30 frames
  - GlobalAveragePooling1D()
- **Top / head:**
  - Dense(256, activation="relu")
  - Dropout(0.3)
  - Dense(num_classes, activation="softmax")
  - Output: class probabilities
- **Trainability pattern**
  - Backbone stays frozen (trainable=False); only the head trains.
- **Diagram:**

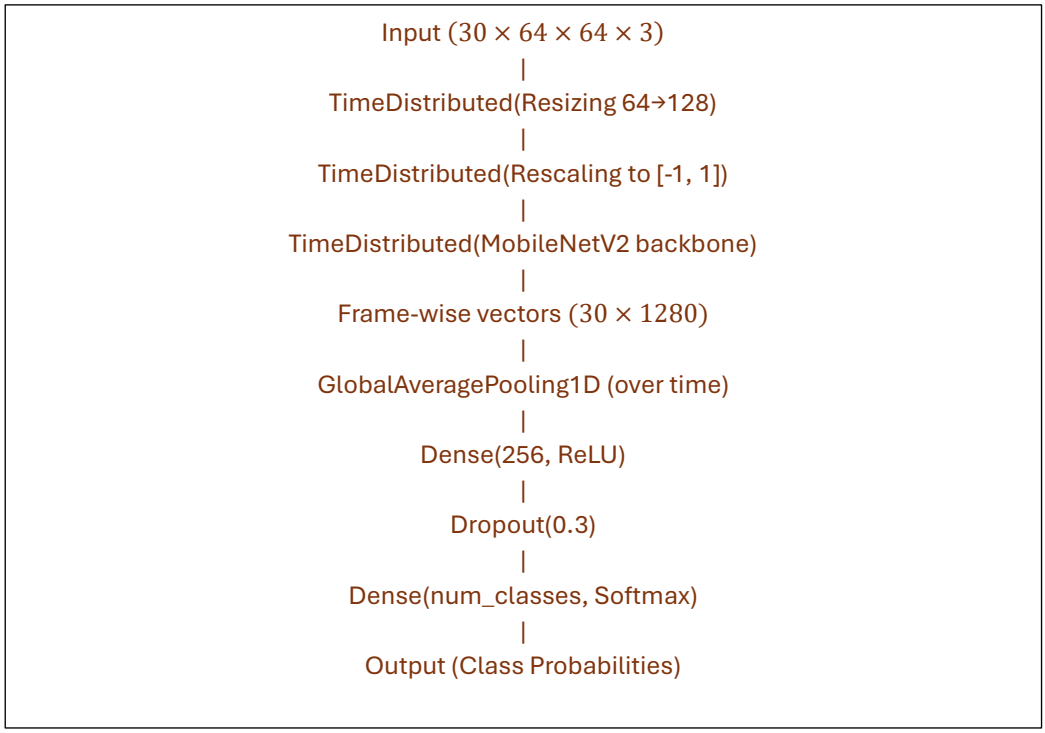


4. Model 4: MobileNetV2

- **Base:** MobileNetV2 (ImageNet weights).
- **Focus**: Lightweight efficiency for mobile devices.
- **Model structure**

- Input: video clip : $(30, 64, 64, 3)$
- Per-frame preprocessing:
  - TimeDistributed (Resizing(IMG_SIZE=128, 128))
  - TimeDistributed (Rescaling(2.0, offset=-1.0)) *(maps* $[0,1] \rightarrow [-1,1]$*)*
- Applied as TimeDistributed(base_model) to each frame
- Temporal aggregation: GlobalAveragePooling1D() across the 30 frame-vectors
- Output: class probabilities

- **Top / head:**
  - Dense(256, activation="relu")
  - Dropout (0.3)
  - Dense (num_classes, activation="softmax")
- **Trainability / fine-tuning pattern**
  - **Phase 1**: base_model.trainable = False (backbone fully frozen)
  - **Phase 2**: base_model.trainable = True
- **Diagram**:



4. Evaluation Metrics:

- **Accuracy**: Overall correct classification percentage.
- **Confusion Matrix**: Visualization of specific misclassifications.
- **Classification Report:** Precision, Recall, and F1-Score.
- **ROC & AUC Curves:** Evaluation of discrimination capability.

5.Comparsion

Model Architecture	*Training Strategy*	*Input Resolution*	*Val/Test Accuracy*	*Model Status*
ResNet50	Transfer Learning (ImageNet)	256 x 256	~80 - 81%	Highest Accuracy
MobileNetV2	Transfer Learning (ImageNet)	128 x 128	78.26%	Most Efficient
Inception V1	Transfer Learning (TF Hub)	224 x 224	75.65%	Moderate Performance
VGG19	Training from Scratch	224 x 224	20.00%	Failed to Converge

5.1. Conclusion & Selection

1. **Deployment Choice: MobileNetV2** is selected as the production model. Its lightweight architecture ensures low latency during real-time inference while maintaining high classification reliability.
2. **Benchmark Choice: ResNet50** serves as the accuracy benchmark, demonstrating the upper limit of performance extractable from the current dataset using standard transfer learning techniques.

6. References

1. **VGG-19 model:**
    - *Paper Title*: "Very Deep Convolutional Networks for Large-Scale Image Recognition"
    - *Authors: Karen Simonyan, Andrew Zisserman (University of Oxford).*
    - *Published In: International Conference on Learning Representations (ICLR),* 2015.
2. **ResNet50:**
    - *Paper:* "Deep Residual Learning for Image Recognition"
    - *Authors:* Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun (2016).
    - *Code Documentation:* Keras ResNet50 API Docs
3. **MobileNet:**
    - *Paper:* "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications"
    - *Authors:* Andrew G. Howard et al. (2017).
    - Code Documentation: *Keras MobileNet API Docs*
4. **Inception V1 (GoogLeNet):**
    - *Paper:* "Going Deeper with Convolutions"

- *Authors:* Christian Szegedy et al. (2015).
- Code Documentation: *Keras MobileNet API Docs*

5. **Dataset Source:** Kaggle: Hand Gesture Recognition Dataset.