Preliminary Documentation for Gesture Recognition Project

Authors: Bora ILCI & Kaan Emre KARA | Group: 10 | Instructor: Valeriya Khan | Date: May 9, 2025

1. Project Overview

Objective: Develop and evaluate a neural-network–based model for hand-gesture recognition, benchmarked against existing state-of-the-art methods.

Scope: Classify 10 static/dynamic hand gestures using open-source datasets (*LeapGestRecog*), with potential augmentation from secondary sources.

2. Algorithm Selection

We will explore and compare the following algorithms:

Algorithm	Description	Example Architecture / Reference
CNN + LSTM	2D CNN extracts spatial features per frame; LSTM captures temporal dynamics over embeddings.	CNN (3×3 kernels; [32,64,128] channels) → FC to 256-d vector → LSTM (128 units) → Dense Softmax Donahue et al. (CVPR'15)
3D-CNN	3D convolutions over spatiotemporal volumes to learn joint features.	C3D: 8 conv layers (3×3×3 kernels) + 2 FC + Softmax Tran et al. (ICCV'15)
Transfer Learning	Pretrained ResNet-18 extracts frame embeddings; temporal pooling or LSTM over them.	ResNet-18 backbone → Global Avg Pool → Dense Softmax He et al. (CVPR'16)

Chosen baseline for Phase 1: CNN+LSTM (PyTorch), with 3D-CNN comparison.

3. Dataset Selection & Description

Primary Dataset: LeapGestRecog

- **Source:** GTI–UPM Gesture Dataset on Kaggle
- **Content:** 10 gesture classes; ~2,000 sequences; ~100 grayscale frames @84×84 each
- **Preprocessing:** Extract frames; resize to 64×64; normalize to [0,1]; folder-per-class

Secondary Dataset (Optional): SHREC Hand Gesture

(Depth + RGB modalities for future multimodal extension)

4. Library & Tool Selection

- **Framework:** `PyTorch 1.13`
- **Preprocessing:** OpenCV, NumPy, pandas
- **Data Utilities:** torch.utils.data.Dataset, DataLoader; torchvision.transforms
- **Visualization & Metrics:** Matplotlib, Seaborn, scikit-learn, TensorBoard
- **Code Quality:** PEP 8 compliance; flake8 linting; docstrings & comments

5. Experimental Plan

1. **Data Preparation:** 70/15/15 train/val/test split; ensure class balance.
2. **Model Implementations:** Baselines A: CNN+LSTM; B: 3D-CNN; C (if time): Transfer Learning.
3. **Training Protocol:** 30 epochs; batch size 16; Adam (LR 1e−3); CrossEntropyLoss; LR scheduler; early stopping (patience 5).
4. **Hyperparameter Tuning:** LR {1e−2,1e−3,1e−4}; hidden sizes {64,128,256}; batch sizes {16,32}.
5. **Evaluation:** Select best model by validation F1; test on held-out set; record metrics.

6. Methods of Result Visualization

- Learning curves: loss & accuracy vs. epochs (Matplotlib)
- Confusion matrix heatmap (Seaborn)
- Per-class metrics bar chart: precision, recall, F1
- Sample sequence predictions: overlay labels on video frames
- Training time & model size table

7. Definition of Quality Measures

- **Accuracy:** Correct classifications / total samples
- **Precision:** TP / (TP + FP) per class
- **Recall:** TP / (TP + FN) per class
- **F1-Score:** 2×(Precision×Recall)/(Precision+Recall)
- **Confusion Matrix Analysis:** Identify common errors
- **Efficiency:** Parameter count, FLOPs, inference time

8. Phase 1 Deliverables

- Written PDF report with sections 1–7
- 5–7 slide deck for consultation (May 12), covering all above
- Discussion points: preprocessing, architectures, computational requirements