



# ASL HAND GESTURE RECOGNITION

Using MediaPipe Landmarks + MLP

2702315223 – Supandi

2702322853 – Evan F Hartono

---

# BACKGROUND

- Deaf users face communication barriers with non-sign-language speakers
- ASL alphabet recognition is a practical first step for sign-to-text systems
- Pixel-based CNNs are heavy; landmark-based models are faster
- Research goal: test practicality + real-time performance on consumer hardware
- MediaPipe provides robust pretrained hand pose estimation



Goal: build a lightweight  
real-time ASL alphabet  
recognition system



Approach: MediaPipe  
Hands (pretrained model)  
+ Multilayer Perceptron



Output: real-time gesture  
prediction from webcam  
input



Framework: FastAPI +  
OpenCV

---

---

# HOW MEDIAPIPE WORKS

## 1. Palm Detection (SSD-Lite CNN)

- Detects palm region, easier than detecting full hand
- Provides bounding box + orientation

## 2. ROI Alignment

- Hand crop normalized to fixed orientation
- Stabilizes input → higher landmark accuracy

## 3. Landmark Regression Model (MobileNetV3-like CNN)

- Predicts 21 landmarks (x, y, z) directly
- Trained on millions of annotated hand poses
- Learns geometry (contours/shape), not skin color

## 4. Tracking + Temporal Smoothing

- Uses previous frame to skip heavy detection
- Enables 20–30 FPS on CPU

# DATASET & FEATURES

Dataset: ASL A–Z + “space”, “nothing”

Balanced at  $\pm 2000$  samples each

Preprocessing:

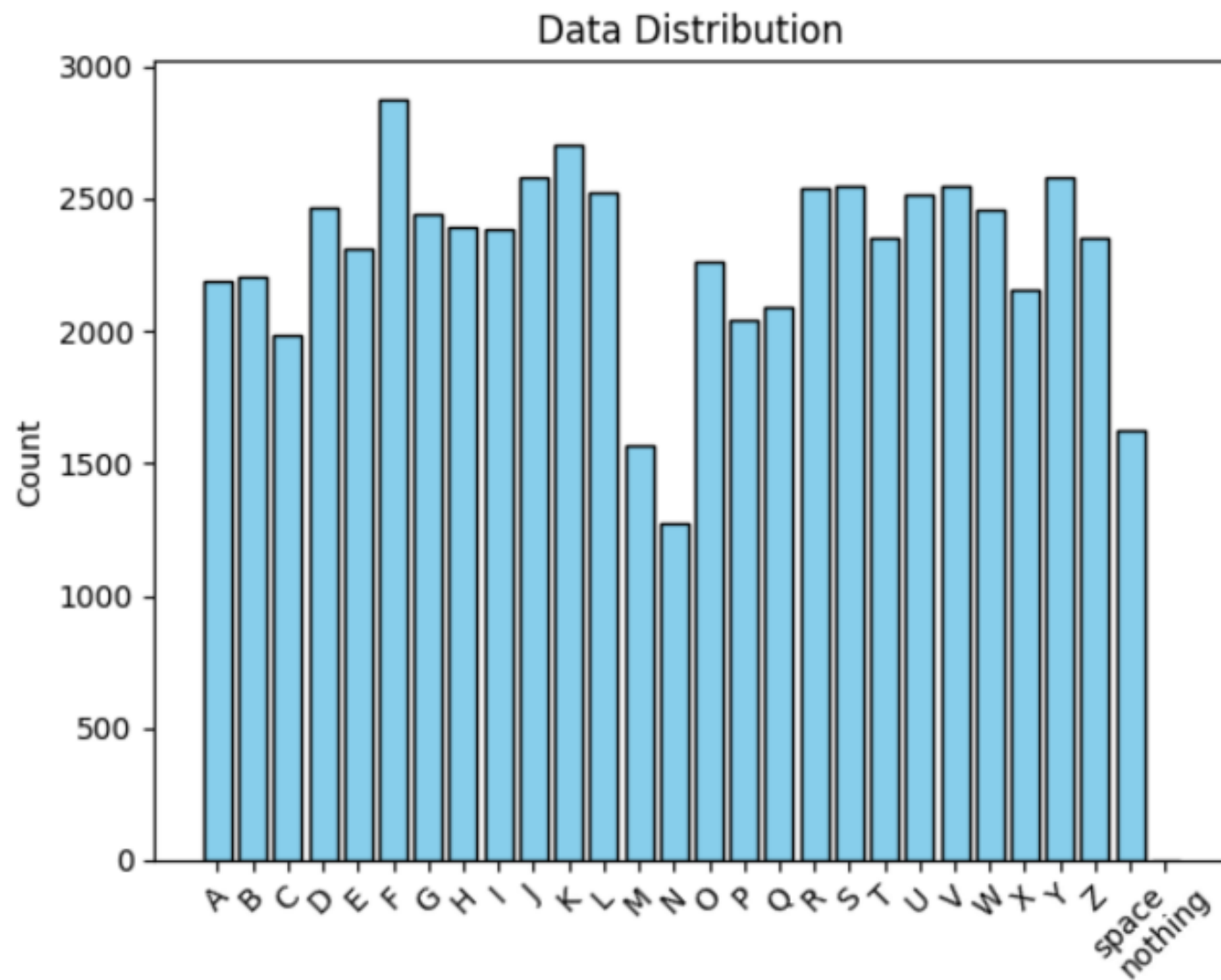
- Extract 21 landmarks  $\rightarrow$  63D feature vector
- Standardization (mean, std)
- Label encoding for gesture classes

Split: 80% train, 10% val, 10% test



---

# CLASS DATASET DISTRIBUTION



---

# MODEL & TRAINING

**Multilayer Perceptron (MLP)**

Input: 63 landmark features

Hidden layers: ReLU

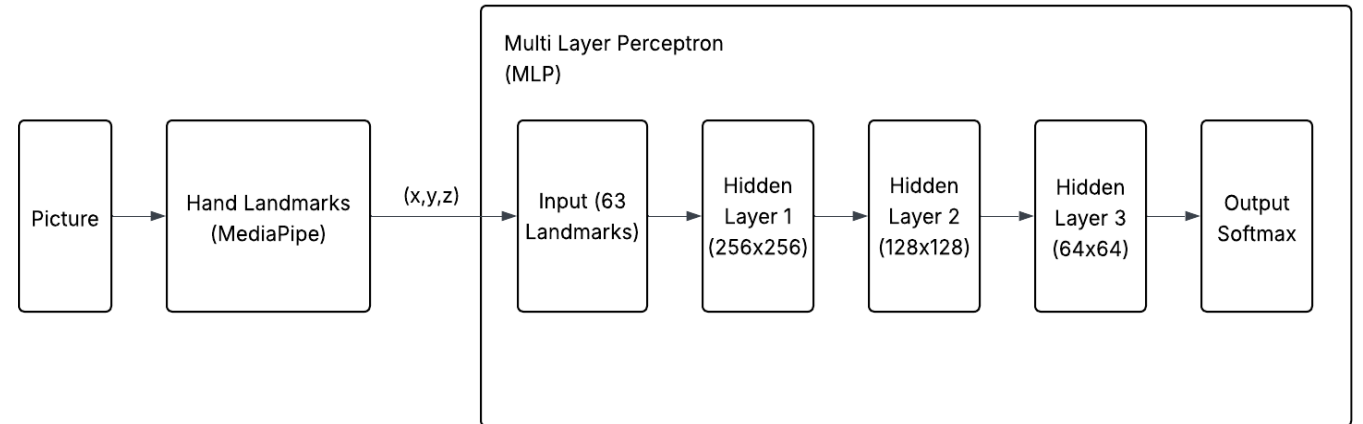
Output: 28 classes

Training: batch 32, early stopping, LR scheduling

**Results:**

**Test Accuracy: 99.23%**

**Test Loss: 0.033**



---

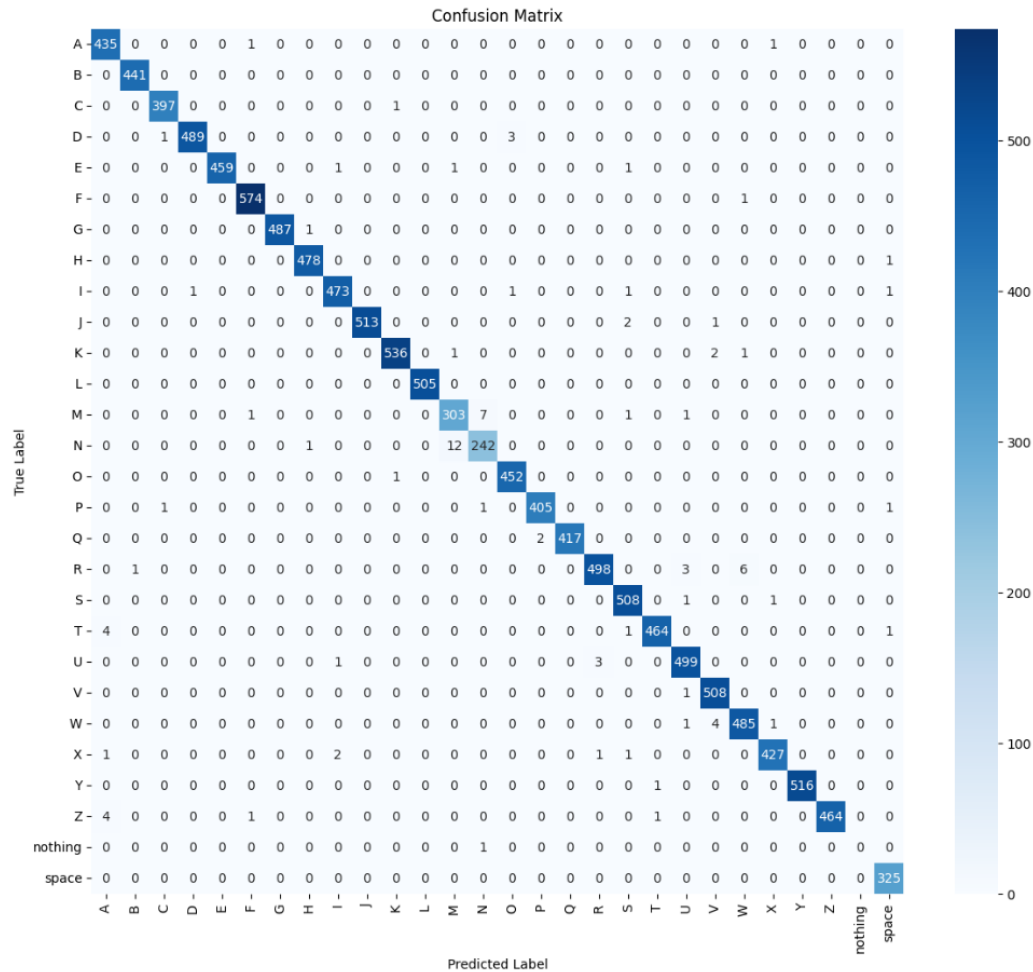
# TRAINING CONFIGURATION

Component	Value
Data training	X_train, y_train
Batch size	32
Epochs (maks.)	100
Validation data	(X_val, y_val)
Callbacks	callbacks (lihat tabel di bawah)
Verbose	1

Callback	Parameter	Nilai
EarlyStopping	monitor	val_loss
	patience	15
	restore_best_weights	True
ReduceLROnPlateau	monitor	val_loss
	factor	0.5
	patience	10
	min_lr	1e-7

---

# EVALUATION

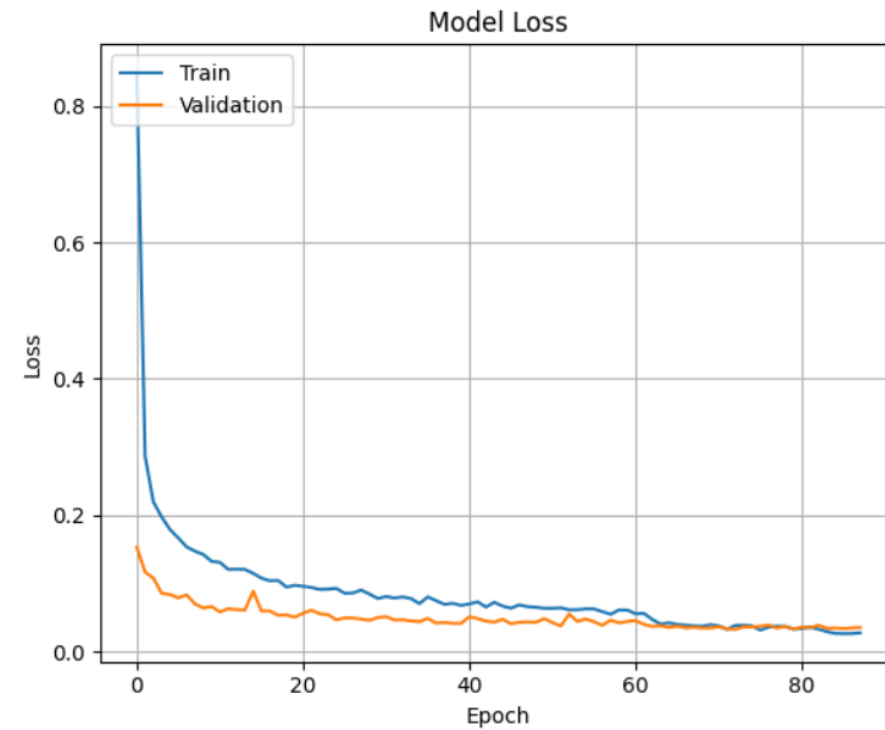
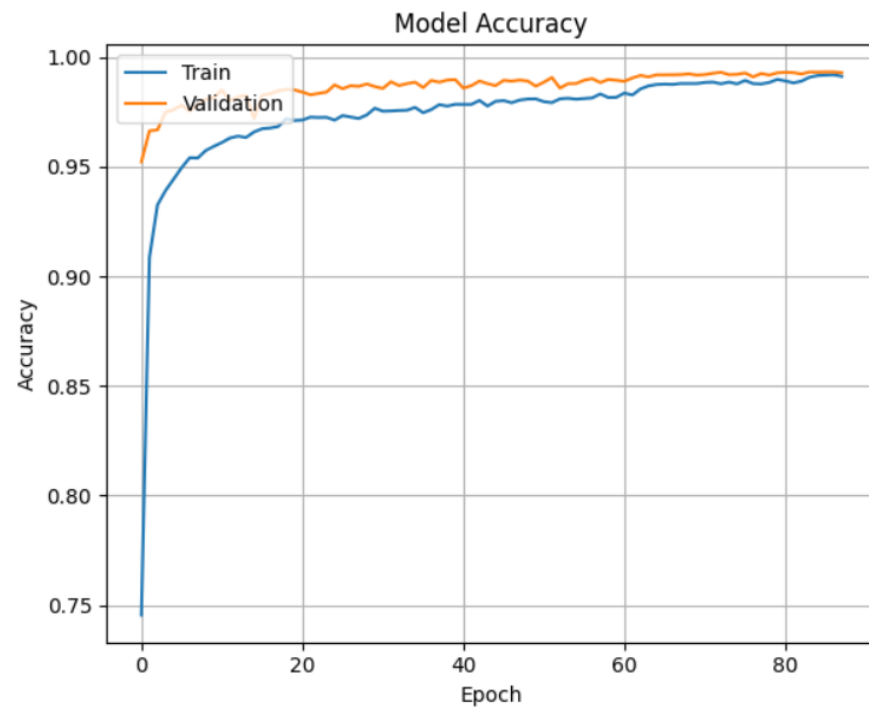


- **Overall pattern:** the matrix is strongly **diagonal**, meaning the model is **highly accurate** for most classes (most off-diagonal values are 0 or very small).
- Best performing classes: **F, L, S, Y** (high confidence, perfect or near-perfect).
- Most confused pair: **M ↔ N** (visually similar gestures or data overlap).
- Weakest class: “nothing” (likely due to class imbalance or labeling issue).



---

# EVALUATION (ACCURACY & LOSS)



---

# EVALUATION & REAL-TIME RESULTS

## **Strengths:**

- Responsive predictions (<1s stabilization)
- Smooth video stream
- Reliable detection in normal lighting & simple backgrounds

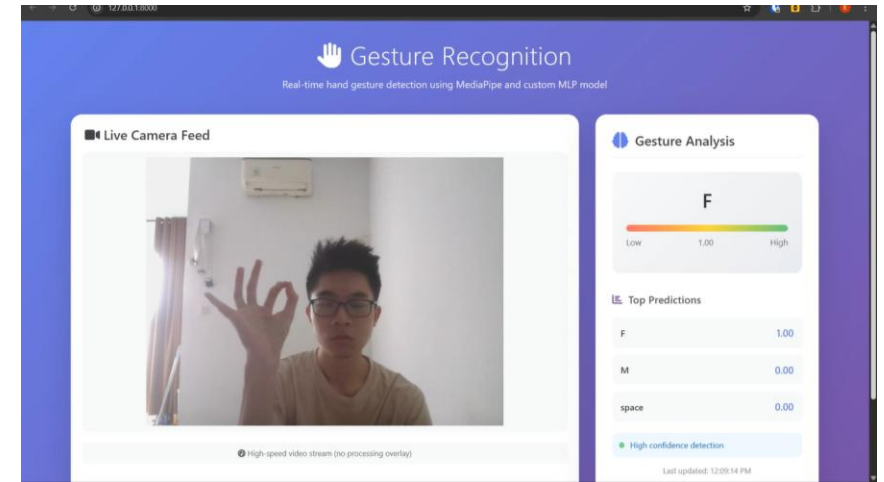
## **Limitations:**

- Lower accuracy in poor lighting
  - Dataset trained only on left hand → right-hand accuracy drops
  - MediaPipe is main CPU bottleneck at high FPS
  - Angle/rotation variance impacts landmark stability
-

---

# SYSTEM APP IMPLEMENTATION

- Framework: **FastAPI** + **OpenCV**
  - Two-thread architecture:
    - Thread 1: high-FPS webcam streaming
    - Thread 2: MediaPipe + MLP prediction
  - Shared state: gesture, confidence, top-3 predictions
  - Temporal smoothing reduces flicker
  - UI designed for minimal latency
- 



```
1 import cv2
2 import mediapipe as mp
3 import numpy as np
4 import joblib
5 from collections import deque
6 from fastapi import FastAPI
7 from fastapi.responses import StreamingResponse, HTMLResponse
8 from datetime import datetime
9 import threading
10 import time
11 import queue
12
13 app = FastAPI()
14
15 video_camera = cv2.VideoCapture(0)
16 prediction_camera = cv2.VideoCapture(0)
17
18 latest_prediction = {
19     "gesture": "Waiting...",
20     "confidence": 0.0,
21     "top_predictions": [],
22     "timestamp": None
23 }
24
25 class CustomGestureDetector:
26     def __init__(self, model_path: str, custom_model_path: str):
27         self.model_data = joblib.load(model_path)
28         self.model = self.model_data['model']
29         self.label_encoder = self.model_data['label_encoder']
30         self.scaler = self.model_data['scaler']
31
32         self.mp_hands = mp.solutions.hands
33         self.hands = self.mp_hands.Hands(
34             static_image_mode=True,
35             max_num_hands=1,
36             min_detection_confidence=0.7,
37             min_tracking_confidence=0.5
38         )
39
40         self.prediction_history = deque(maxlen=5)
41         self.confidence_threshold = 0.6
42
```

---

# REFLECTION & NEXT STEPS

## **What worked:**

- Landmark-based pipeline extremely lightweight
- High offline accuracy with simple MLP
- Real-time performance usable with threading

## **Issues found:**

- Lighting + orientation sensitivity
- Left-hand dataset bias
- MediaPipe jitter in non-ideal environments

## **Improvements:**

- Add mirrored right-hand data
  - Augment lighting/angle variations
  - Consider hybrid model: landmarks + image features
  - Migrate MediaPipe to GPU/ONNX for speed
-