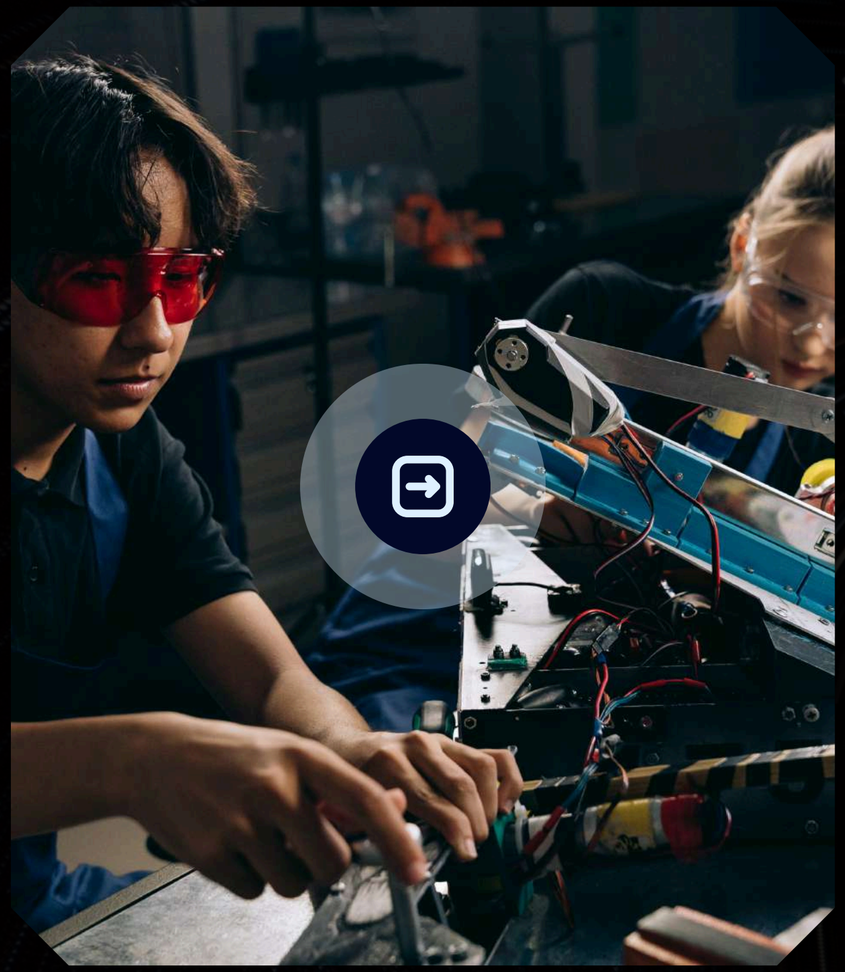
Deep Learning Project - LB01

AMERICAN SIGN LANGUAGE RECOGNITION

Chang Minfang – 2702335686
Sherlyn Usin – 2702253894



INTRODUCTION

Problem Background

Despite advances in ASL recognition, practical deployment remains challenging. Many high-accuracy models are computationally intensive, limiting their use in real-time applications on consumer devices. In addition, subtle differences between similar ASL hand gestures, such as ‘M’, ‘N’, and ‘T’, increase the risk of misclassification. These challenges are further compounded by varying lighting conditions and background noise in live webcam environments, highlighting the need for a lightweight yet robust ASL recognition system.

LITERATURE REVIEW

American Sign Language Recognition

ASL is a visual language using hand shapes and gestures for communication

Deep Learning for Image-Based Gesture Recognition

CNNs automatically extract spatial features and outperform traditional methods

Transfer Learning in ASL Recognition

Pre-trained models improve accuracy and reduce training time

Real-Time ASL Recognition Systems

Real-time systems require ROI extraction and prediction smoothing

ASL Datasets

Public datasets enable supervised training and fair evaluation

DATASET

```
Memuat gambar...
  [A] 400 gambar
  [B] 400 gambar
  [C] 400 gambar
  [D] 400 gambar
  [DELETE] 400 gambar
  [E] 400 gambar
  [F] 400 gambar
  [G] 400 gambar
  [H] 400 gambar
  [I] 400 gambar
  [J] 400 gambar
  [K] 400 gambar
  [L] 400 gambar
  [M] 400 gambar
  [N] 400 gambar
  [O] 400 gambar
  [P] 400 gambar
  [Q] 400 gambar
  [R] 400 gambar
  [S] 400 gambar
  [SPACE] 400 gambar
  [T] 400 gambar
  [U] 400 gambar
  [V] 400 gambar
  [W] 400 gambar
  [X] 400 gambar
  [Y] 400 gambar
  [Z] 400 gambar

Total gambar: 11200
```

Images of American Sign Language (ASL) Alphabet Gestures (Mendeley Data)

- Raw Gesture Data Type
- 400 images for each class
- Total dataset : ~11200 images
- Number of classes : 28 (A–Z, DELETE, SPACE)
- Click here to view the dataset

Architecture and Technology Used

Framework
Tensorflow/Keras

Model
MobileNetV2 (Transfer Learning) with Two-Phase Training (Fine-Tuning)

Frontend
Streamlit dan streamlit-webrtc (WebRTC API)

- In this project, we used TensorFlow/Keras as the primary framework to build and train deep learning models. The model used was MobileNetV2, which uses a transfer learning approach, which allows the use of pre-trained weights for more efficient and accurate training, especially with limited data. The training process was carried out in two stages: initial training and fine-tuning to improve model performance.
- For the interface, we built the application using Streamlit combined with streamlit-webrtc (WebRTC API), enabling the system to capture video in real time and display ASL gesture prediction results directly to the user.

Custom Classification Head

A sequence of layers was added to adapt the learned features to the 28-class ASL task

- **Global Average Pooling 2D (GAP)**: Reduces the feature map dimensionality.
- **Batch Normalization**: Included to stabilize training and accelerate convergence.
- **Dropout (0.5)**: A high dropout rate to prevent initial overfitting.
- **Dense Layer (256 units, ReLU)**: For deep feature processing.
- **Batch Normalization**: Introduced to stabilize activations after the second feature transformation.
- **Dropout (0.3)**: A secondary dropout layer to further improve generalization.
- **Output Layer**: A final dense layer with 28 units and a Softmax activation function, corresponding to the probability distribution over the 28 class indices.

Training Setup

Two-phase transfer learning strategy to adapt ImageNet features to ASL gestures while minimizing catastrophic forgetting

Phase 1 – Feature Extraction

MobileNetV2 backbone frozen; only the custom classification head trained for 30 epochs using Adam optimizer (LR = 0.001)

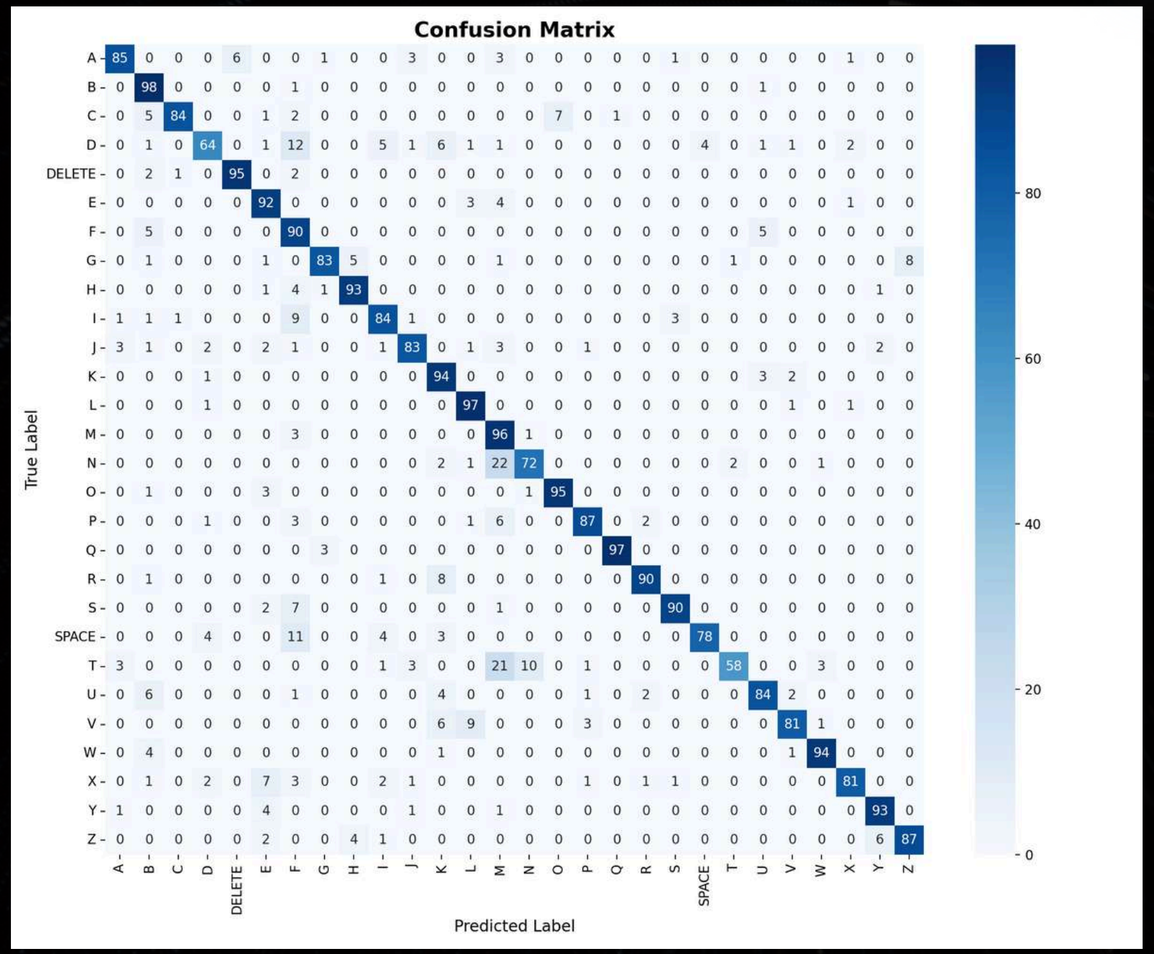
Phase 2 – Fine-Tuning

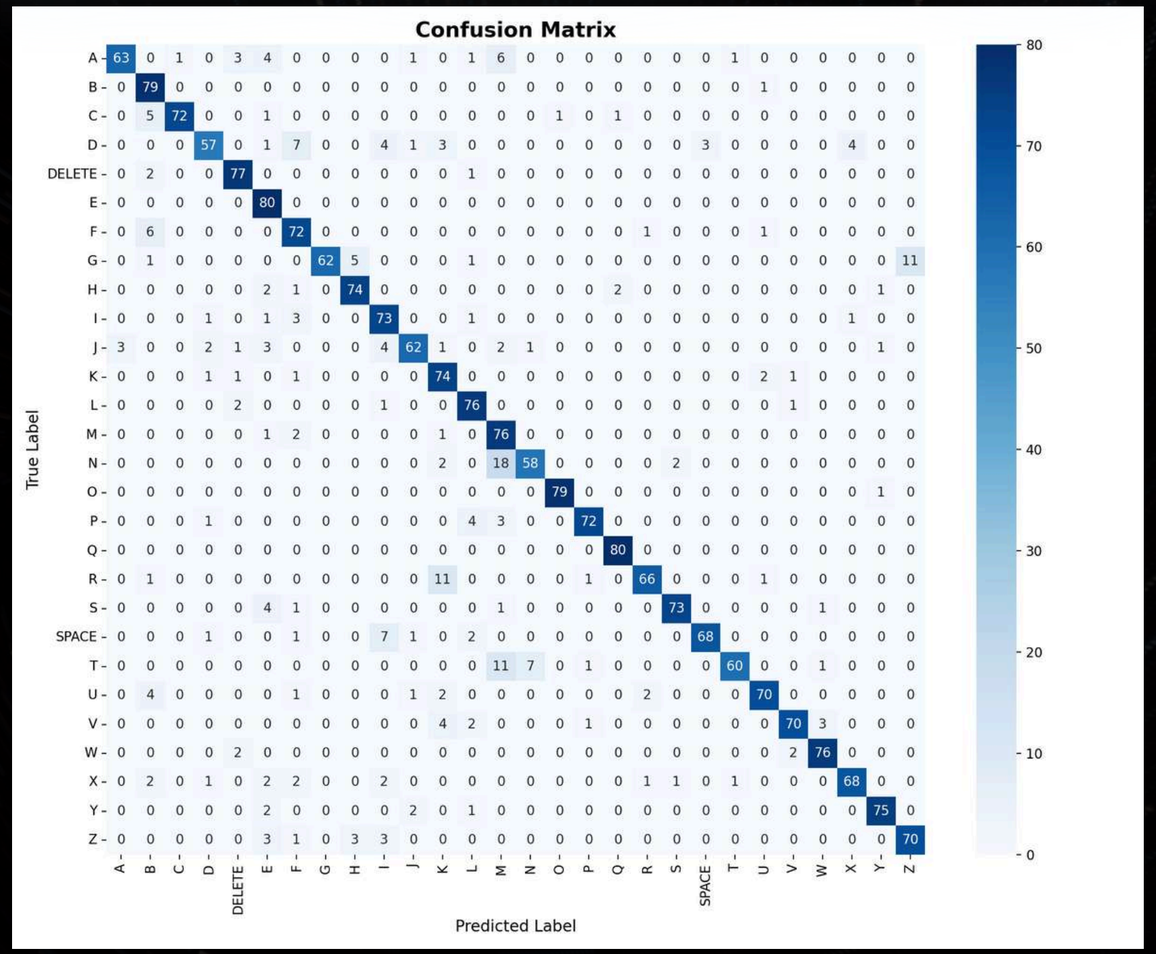
Top 30 convolutional layers unfrozen and trained jointly with the classification head for 20 epochs (LR = 0.0001)

The model was trained using a two-phase transfer learning strategy to adapt ImageNet features to ASL gestures while preventing catastrophic forgetting. Initial training focused on the custom classification head with frozen backbone layers, followed by selective fine-tuning of the top convolutional layers using a lower learning rate. A coarse-to-fine optimization schedule ensured stable convergence, while Dropout regularization improved generalization. Sparse categorical cross-entropy and macro F1-score were used to ensure balanced multi-class performance.

EVALUATION

Confusion Matrix





The Confusion Matrix shows model performance in detail per class. The main diagonal (from top left to bottom right) represents the number of correct predictions (True Positives).

EVALUATION

Classification Report (Test)

```
Classification Report (Test):
```

	precision	recall	f1-score	support
A	0.9140	0.8500	0.8808	100
B	0.7717	0.9800	0.8634	100
C	0.9767	0.8400	0.9032	100
D	0.8533	0.6400	0.7314	100
DELETE	0.9406	0.9500	0.9453	100
E	0.7931	0.9200	0.8519	100
F	0.6040	0.9000	0.7229	100
G	0.9432	0.8300	0.8830	100
H	0.9118	0.9300	0.9208	100
I	0.8485	0.8400	0.8442	100
J	0.8925	0.8300	0.8601	100
K	0.7581	0.9400	0.8393	100
L	0.8584	0.9700	0.9108	100
M	0.6038	0.9600	0.7413	100
N	0.8571	0.7200	0.7826	100
O	0.9314	0.9500	0.9406	100
P	0.9255	0.8700	0.8969	100
Q	0.9898	0.9700	0.9798	100
R	0.9474	0.9000	0.9231	100
S	0.9474	0.9000	0.9231	100
SPACE	0.9512	0.7800	0.8571	100
T	0.9508	0.5800	0.7205	100
U	0.8936	0.8400	0.8660	100
V	0.9205	0.8100	0.8617	100
W	0.9495	0.9400	0.9447	100
X	0.9419	0.8100	0.8710	100
Y	0.9118	0.9300	0.9208	100
Z	0.9158	0.8700	0.8923	100
accuracy			0.8661	2800
macro avg	0.8823	0.8661	0.8671	2800
weighted avg	0.8823	0.8661	0.8671	2800

Test Results

```
Test Results:
  Loss: 0.5121
  Accuracy: 0.8661 (86.61%)
```

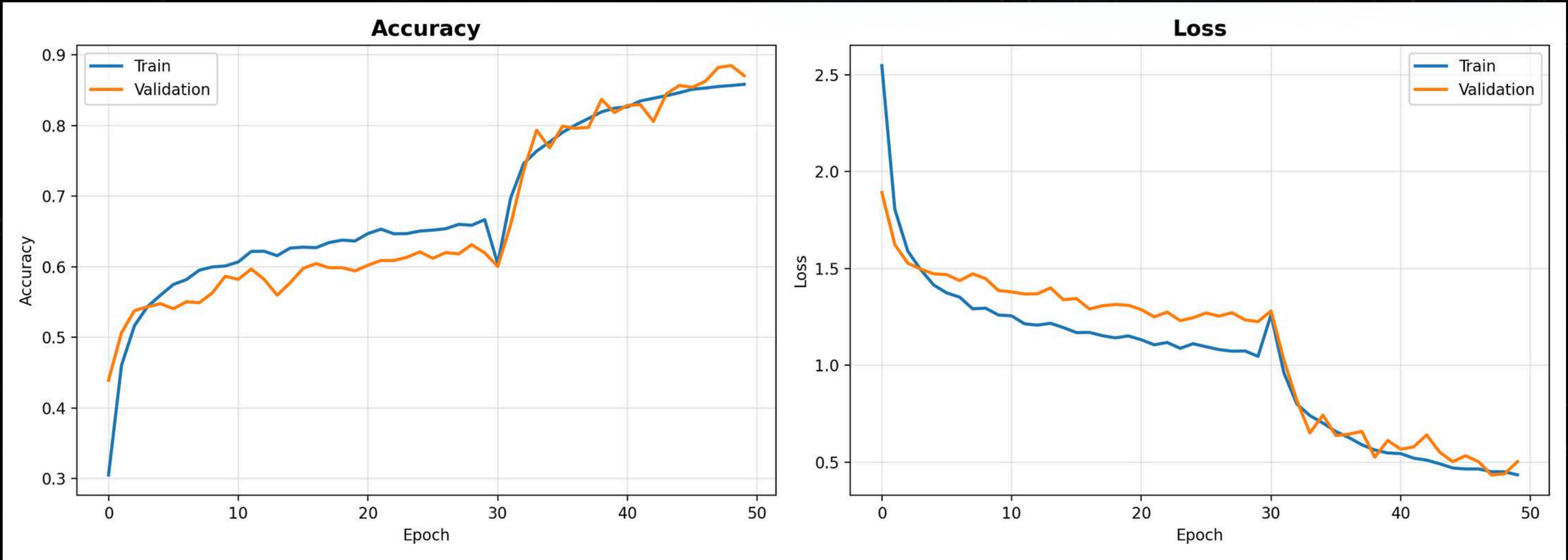
Accuracy

```
Final Accuracy:
  Validation: 88.48%
  Test: 86.61%
```

- In the evaluation phase, the model's performance was tested using test data not seen during the training process. The test results showed a test accuracy of 86.61% with a loss of 0.5121, indicating the model's ability to recognize ASL gestures quite well.
- The validation accuracy reached 88.48%, indicating that the model did not experience significant overfitting and had good generalization capabilities.
- The classification report showed relatively stable precision, recall, and F1-score values across each ASL letter class, enabling the model to consistently recognize most gestures.

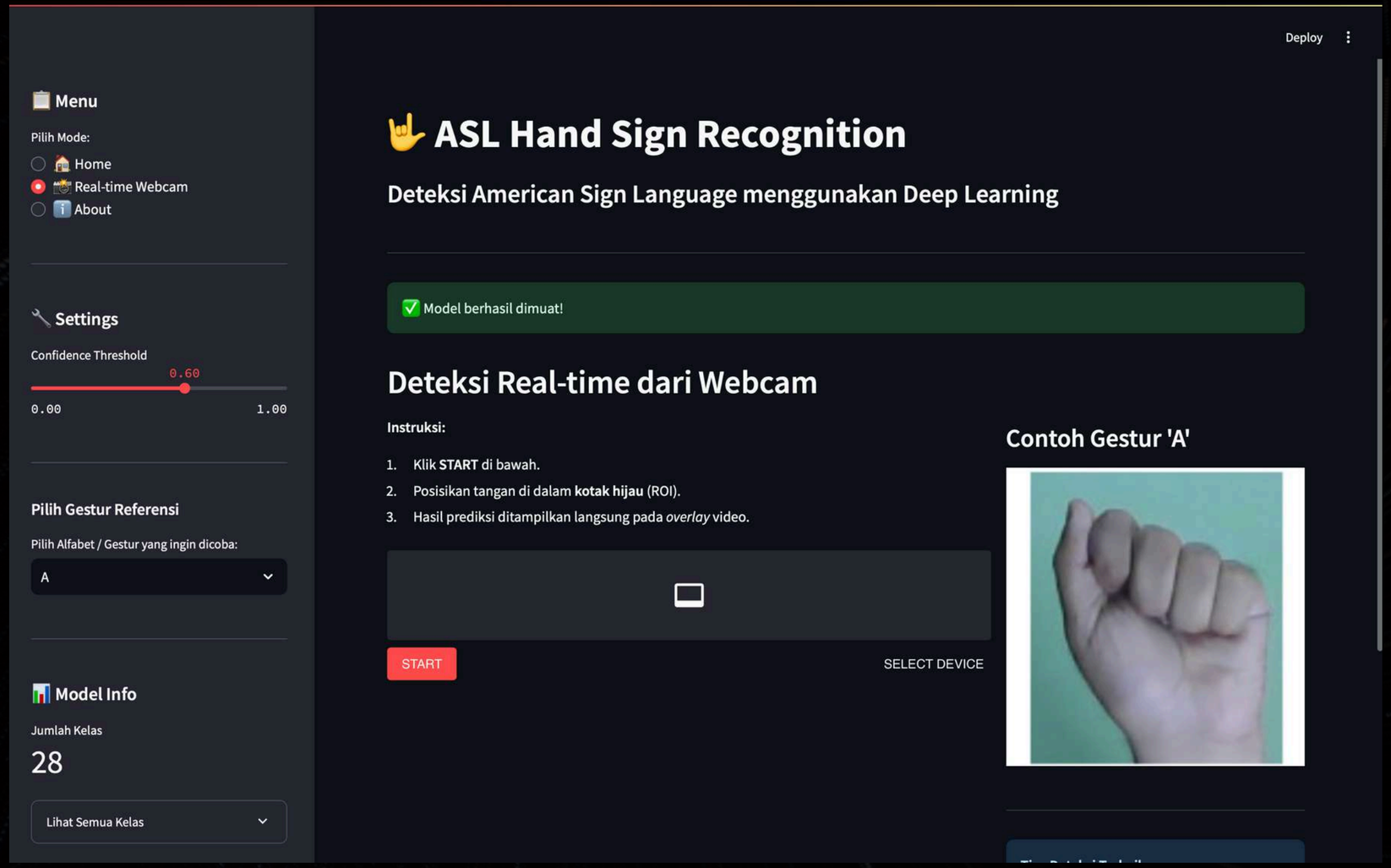
EVALUATION

Training History



The graph shows the development of accuracy and loss during the training and validation processes. Training and validation accuracy consistently increase with each epoch, while the loss decreases, indicating successful model learning.

The gap between the training and validation curves is relatively small, indicating that the model does not experience significant overfitting. The significant performance increase at certain epochs demonstrates the effect of fine-tuning the transfer learning model.f



DEMO

Click here to see our demo

```
"""Create CNN model for ASL recognition"""
model = keras.Sequential([
    # First convolutional block
    keras.Input(shape=input_shape),
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    # Second convolutional block
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    # Third convolutional block
    layers.Conv2D(128, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),
    layers.Dropout(0.25),

    # Flatten and dense layers
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation='softmax')
])
```

REFLECTION

Before using MobileNetV2, we initially experimented with a manually designed CNN model. However, the performance was not satisfactory, especially in handling visually similar ASL gestures and maintaining stability in real-time inference. Through this experience, we learned that pre-trained architectures with transfer learning, such as MobileNetV2, can significantly improve feature extraction and generalization, particularly when working with limited datasets and deployment constraints. This comparison helped us better understand the strengths and limitations of custom CNN models versus pre-trained architectures.

LINK

Link to Drive

Link to Git

THANK YOU

https://github.com/keziacminf/DeepLearning_ASL