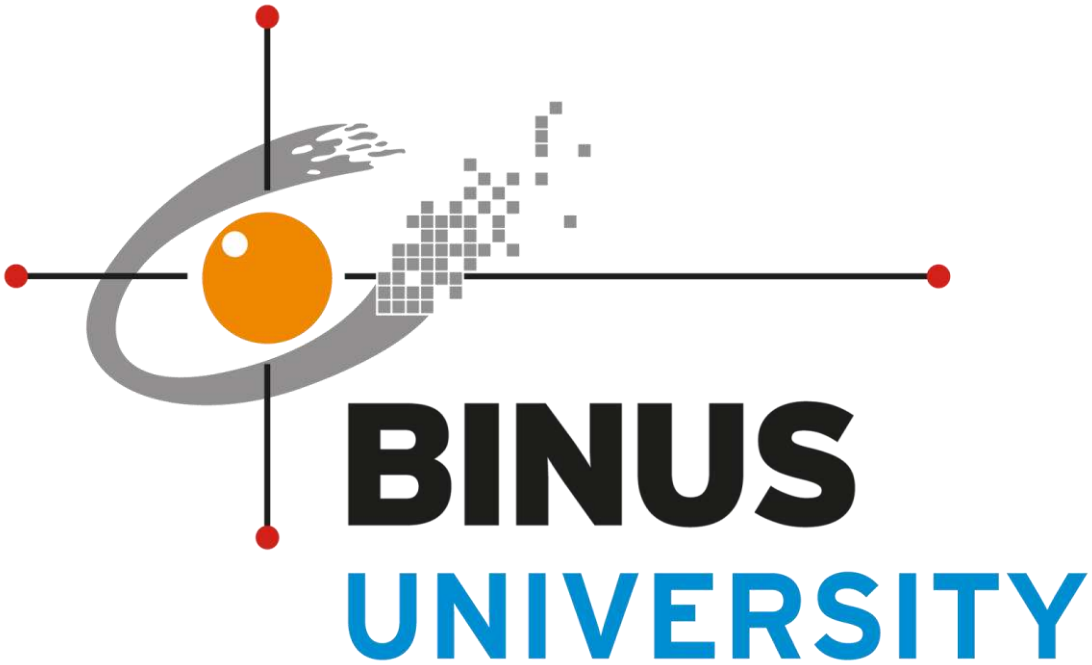
LAPORAN AKHIR *PROJECT DEEP LEARNING*

Real-Time American Sign Language (ASL) Alphabet Recognition System using MobileNetV2 Architecture



Disusun oleh :

1. Chang Minfang 2702335686
2. Sherlyn Usin 2702253894

UNIVERSITAS BINA NUSANTARA

JAKARTA

2025

1. Introduction

1.1. Background

The rapid progress of Computer Vision (CV) has significantly reshaped human–computer interaction, particularly in the development of assistive technologies for people with disabilities. Within the Deaf and Hard of Hearing (DHH) community, American Sign Language (ASL) functions as a primary means of communication. However, limited ASL proficiency among the general population continues to create substantial communication barriers (Rastgoo et al., 2020). As emphasized by Sowrow et al. (2024), the availability of reliable automated recognition systems plays a crucial role in reducing these barriers and fostering more inclusive social and professional interactions.

At the same time, advances in Deep Learning—especially Convolutional Neural Networks (CNNs)—have achieved strong performance in image classification tasks. For real-world deployment, sign language recognition systems must be capable of running efficiently on mobile or edge devices with constrained computational resources. In response to this requirement, this study adopts the MobileNetV2 architecture, which leverages inverted residuals and linear bottlenecks to preserve high recognition accuracy while substantially lowering parameter count and inference latency (Sandler et al., 2018).

1.2. Problem Statement

Although numerous ASL recognition models have been proposed, several fundamental challenges remain unresolved. Many approaches that achieve high accuracy rely on computationally intensive architectures, limiting their feasibility for real-time deployment on standard consumer devices (Sandler et al., 2018). In addition, ASL alphabet gestures often differ only in subtle finger positions and hand configurations—such as the letters *M*, *N*, and *T*—which demands highly precise feature extraction to prevent misclassification (Sowrow et al., 2024). Beyond model accuracy, practical deployment also requires robustness to real-world conditions, including variations in lighting and background complexity commonly encountered in live webcam settings.

1.3.Objectives

The primary objectives of this project are:

1. To construct a robust classification model for ASL alphabet gestures using the MobileNetV2 architecture with transfer learning.
2. To evaluate the model's performance on the Mendeley ASL Alphabet Gestures benchmark dataset using metrics such as accuracy, F1-score, and confusion matrix.
3. To develop and deploy a real-time detection application that integrates the trained model with a webcam interface for live inference.

1.4. Scope

This project addresses the classification of 28 distinct gesture classes, encompassing the ASL alphabet (A–Z) as well as the *SPACE* and *DELETE* tokens. Model training is conducted using static image-based data, while real-time recognition incorporates a temporal smoothing strategy to stabilize predictions from continuous video streams. The system is deployed in a local environment and relies on a standard RGB webcam as the primary input device, with gesture detection focused on hand movements within a predefined Region of Interest (ROI).

2. Literature Review

2.1.American Sign Language Recognition

American Sign Language (ASL) is a visual language that conveys meaning through hand shapes, orientations, and movements. Automated ASL recognition has therefore attracted considerable research interest due to its potential to reduce communication barriers between the Deaf community and the broader population (Koller, 2020). Early recognition systems primarily depended on handcrafted features, including hand shape descriptors, edge-based methods, and color segmentation. While effective in controlled settings, these approaches were highly sensitive to changes in lighting conditions and background complexity, which limited their reliability in real-world scenarios (Starner & Pentland, 1997).

The emergence of deep learning, particularly Convolutional Neural Networks (CNNs), has led to significant improvements in both accuracy and robustness for ASL recognition tasks. CNN-based models are capable of automatically learning discriminative spatial features directly from image data, thereby reducing reliance on manual feature engineering (LeCun et al., 2015). Empirical studies have consistently shown that CNNs outperform traditional machine learning methods in the recognition of static ASL alphabet gestures (Oyedotun & Khashman, 2017).

2.2.Deep Learning for Image-Based Gesture Recognition

Convolutional Neural Networks (CNNs) have become the standard architecture for image classification because of their ability to model spatial hierarchies within visual data. A typical CNN is composed of convolutional, pooling, and fully connected layers that progressively extract features ranging from simple edges to more abstract representations (Goodfellow et al., 2016). In the context of gesture recognition, CNNs are especially effective at capturing hand contours, finger configurations, and subtle shape variations.

Numerous studies have applied CNN-based approaches to hand gesture and sign language recognition using established benchmark datasets. Simonyan and Zisserman (2015) showed that deeper CNN architectures can substantially improve performance in visual recognition tasks. Nevertheless, increasing network depth often comes at the cost of higher computational and memory requirements, which limits the practicality of such models for real-time applications on resource-constrained devices.

To overcome these challenges, recent research has focused on lightweight CNN architectures that preserve competitive accuracy while significantly reducing computational complexity (Howard et al., 2017). These efficient models are particularly well suited for real-time ASL recognition systems, where low latency and fast inference are critical deployment requirements.

2.3.Transfer Learning in ASL Recognition

Transfer learning has become a widely adopted strategy for improving model performance, particularly in scenarios where labeled training data are limited. This approach involves adapting a model pre-trained on large-scale datasets such as ImageNet to a target task through fine-tuning, enabling the reuse of general visual representations learned from millions of images (Pan & Yang, 2010).

In ASL recognition research, transfer learning has proven effective in accelerating convergence and enhancing classification accuracy. Pre-trained architectures including VGGNet, ResNet, and MobileNet have been successfully applied to ASL alphabet recognition tasks (Aly & Aly, 2020). Beyond reducing training time, transfer learning also improves generalization, especially in datasets that exhibit variations in lighting conditions, hand sizes, and background complexity.

Among these architectures, MobileNetV2 offers a particularly favorable trade-off between accuracy and computational efficiency. By employing depthwise separable convolutions, MobileNetV2 substantially reduces parameter count and floating-point operations, making it well suited for real-time inference on resource-constrained devices (Sandler et al., 2018).

2.4.Real-Time ASL Recognition Systems

Real-time ASL recognition presents challenges that extend beyond offline image classification, including strict latency requirements, camera noise, and temporal prediction instability. To mitigate rapid fluctuations in frame-by-frame predictions, real-time systems commonly employ temporal smoothing strategies such as majority voting or sliding window averaging (Molchanov et al., 2016).

Several studies have demonstrated real-time hand gesture recognition using webcam input in combination with deep learning models. These pipelines typically incorporate a region of interest (ROI) extraction stage, followed by image preprocessing and gesture classification (Zhang et al., 2021). When

paired with efficient CNN architectures, such frameworks enable practical, low-latency ASL recognition on consumer-grade hardware.

Nevertheless, persistent challenges remain, particularly in distinguishing visually similar ASL gestures such as *M* and *N* or *D* and *T*. Accurately resolving these ambiguities often requires high-resolution feature representations and precise hand segmentation, which continue to be active areas of research (Koller, 2020).

2.5. ASL Datasets

The availability of high-quality datasets is fundamental to the development of reliable ASL recognition systems. Public datasets, such as the ASL Alphabet dataset and ASL gesture collections hosted on Mendeley Data, provide labeled images captured under controlled conditions, making them suitable for supervised learning and benchmarking tasks (Sowrow et al., 2024).

In this study, the dataset consists of static images representing ASL alphabet gestures, along with special symbols such as DELETE and SPACE. This structured labeling supports supervised classification and enables consistent evaluation of model performance across multiple gesture classes (Sowrow et al., 2024).

3. Dataset

3.1. Dataset Source

The primary dataset used in this study is the *Images of American Sign Language (ASL) Alphabet Gestures, Version 2*, released by Sowrow et al. (2024) and made available through Mendeley Data (link source dataset: https://data.mendeley.com/datasets/48dg9vhmyk/2). This dataset is selected for its comprehensive representation of the ASL alphabet, making it well suited for training and evaluating robust gesture recognition models.

3.2. Dataset Description

The classification model was trained and evaluated on 28 distinct classes, consisting of the 26 letters of the English alphabet (A-Z) along with two key function gestures: 'SPACE' and 'DELETE'.

- Total Images : The dataset comprises a total of about 11.200 images (about 400 images for each class).
- Number of classes : 28 (A-Z, SPACE, DELETE).
- Image Format : All images are in JPEG (.jpg) format.
- Characteristic : The dataset features clear, isolated hand gestures, focusing on the hand shape and orientation crucial for accurate ASL recognition.

3.3.Preprocessing

A series of robust preprocessing steps were executed to prepare the raw images for the MobileNetV2 architecture:

1. Image Resizing: All input images were resized to a uniform resolution of 224 × 224 pixels to conform to the standard input dimensions of the pre-trained MobileNetV2 base model.
2. Normalization and Preprocessing: Pixel intensity values were processed using tf.keras.applications.mobilenet_v2.preprocess_input, which standardizes the input by transforming pixel values from the original 0–255 range to −1 to 1. This normalization step is required to ensure compatibility with the pre-trained MobileNetV2 base weights.
3. Data Augmentation: To improve the model's generalization ability and robustness to environmental variations, the training data were augmented on the fly using random horizontal flipping, rotation (±15%), zoom (±15%), and contrast adjustment (±20%).
4. Data Split: The dataset was divided into training and validation subsets using an 80:20 ratio with stratified sampling (stratify = y_train) to ensure proportional representation of all 28 classes in both splits. A separate and independent portion of the data was reserved as the test set to provide an unbiased evaluation of the final model performance.

The applied data augmentation techniques are designed to improve the robustness of learned feature representations. Random rotation and zoom simulate natural variations in hand orientation and scale, while contrast adjustment enhances invariance to lighting conditions commonly encountered in real-time webcam usage. Horizontal flipping is applied selectively to

preserve gesture semantics while increasing sample diversity. These augmentations encourage the network to learn invariant and discriminative features rather than memorizing static pixel patterns, thereby improving generalization performance during both validation and real-time inference.

4. Modeling

4.1. Model Architecture

The classification task was approached using a transfer learning strategy built on the MobileNetV2 architecture. MobileNetV2 was selected for its favorable trade-off between classification accuracy and computational efficiency, which makes it well suited for real-time deployment scenarios (Sandler et al., 2018). The main components of the proposed model architecture are outlined as follows:

1. Base Model: The pre-trained weights from the ImageNet benchmark dataset were loaded onto the MobileNetV2 convolutional base.
2. Fine-Tuning Strategy: The initial layers of the base model were kept frozen to retain the general feature extraction capabilities learned from ImageNet. Only the upper-most convolutional blocks were unfrozen to allow for fine-tuning, adapting the weights to the specific features of ASL hand gestures.
3. 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**

      The Global Average Pooling (GAP) layer is applied directly after the final convolutional block of MobileNetV2 to aggregate spatial information across feature maps. Instead of flattening the feature maps, GAP computes the average activation of each feature channel, resulting in a compact feature representation. This approach significantly reduces the number of trainable parameters, mitigates overfitting, and preserves high-level semantic information related to hand shape and finger configuration. Placing GAP immediately after the convolutional

backbone ensures that spatially distributed features are summarized before entering the fully connected layers, which is particularly important for efficient real-time inference.

- **Batch Normalization: Included to stabilize training and accelerate convergence**

  Batch Normalization is introduced after feature aggregation to normalize the distribution of activations. By reducing internal covariate shift, this layer improves gradient flow and stabilizes the optimization process. Its placement directly after GAP ensures that the aggregated feature vector maintains a consistent scale before undergoing regularization and non-linear transformation, which is especially beneficial during fine-tuning of pre-trained networks.

- **Dropout (0.5): A high dropout rate to prevent initial overfitting**

  A Dropout layer with a rate of 0.5 is applied to aggressively regularize the high-dimensional feature representation produced by the convolutional backbone. This early-stage regularization prevents co-adaptation of neurons and reduces the risk of overfitting during the initial training and fine-tuning phases. Applying a higher dropout rate at this stage enforces robustness in the learned representations before deeper feature transformations are performed.

- **Dense Layer (256 units, ReLU): For deep feature processing**

  The Dense layer with 256 units and ReLU activation serves as a non-linear transformation stage that enhances the representational capacity of the model. This layer enables the network to learn complex and discriminative feature combinations necessary to distinguish subtle differences between visually similar ASL gestures (e.g., *M*, *N*, and *T*). The choice of 256 units provides a balance between expressive power and model complexity, allowing effective feature refinement without excessive parameter growth.

- **Batch Normalization: For stabilization of the second transformation layer**

  A second Batch Normalization layer is applied after the dense transformation to further stabilize learning dynamics and maintain consistent gradient propagation. This normalization step helps regulate the scale of activations after the non-linear transformation, improving training stability and ensuring that subsequent regularization operates on well-conditioned feature distributions.

- **Dropout (0.3): Second dropout layer for controlled regularization**

  A Dropout layer with a reduced rate of 0.3 is applied following the second Batch Normalization. The lower dropout rate allows later layers to retain more discriminative information while still providing effective regularization. This progressive reduction in dropout strength reflects a design strategy where stronger regularization is applied to earlier, more abstract representations, while later layers focus on fine-grained class discrimination.

- **Output Layer: Softmax-based multi-class classification**

  The final output layer consists of a Dense layer with 28 units and a Softmax activation function, producing a normalized probability distribution over the 28 ASL class indices (0–27). This formulation enables multi-class classification and is trained using sparse categorical cross-entropy, which is computationally efficient for integer-encoded class labels.

Overall, the custom classification head is carefully designed to balance model capacity, training stability, and generalization, ensuring that the fine-tuned MobileNetV2 architecture can effectively distinguish subtle ASL hand gestures while remaining computationally efficient for real-time inference.

4.2.Training Setup

Model training was conducted using a structured two-phase strategy designed to maximize the benefits of transfer learning while minimizing the risk of catastrophic forgetting of the pre-trained MobileNetV2 weights. This approach allows the model to progressively adapt from generic visual representations to task-specific ASL gesture features.

1. Phase 1: Feature Extraction.

   In the first phase, the MobileNetV2 backbone was initialized with ImageNet pre-trained weights, and all convolutional layers were frozen (trainable = False). Training was limited to the newly added custom classification head and conducted for 30 epochs. The model was compiled using the Adam optimizer with an initial learning rate of 0.001. The objective of this phase was to rapidly adapt the custom layers so they could effectively map the generic visual features learned by MobileNetV2 to the 28 target ASL classes.

2. Phase 2: Fine-Tuning.

   The second phase focused on refined model adaptation. The MobileNetV2 backbone was partially unfrozen, with only the top 30 convolutional layers set as trainable, while the remaining layers remained frozen. These trainable layers were optimized jointly with the custom classification head.

   The model was recompiled using the Adam optimizer with a reduced learning rate of 0.0001 and trained for an additional 20 epochs. The lower learning rate enables gradual weight updates in the pre-trained convolutional filters, allowing the model to capture subtle hand shapes and finger orientations critical for accurate ASL alphabet recognition.

   The decision to unfreeze only the top 30 layers is motivated by the hierarchical nature of convolutional neural networks. Lower layers typically learn generic low-level features such as edges, textures, and simple geometric patterns that are largely task-agnostic, while higher layers encode more task-specific and semantically rich representations. Fine-tuning only the upper convolutional blocks allows the model to

adapt high-level features to ASL-specific hand configurations while preserving robust low-level visual representations learned from ImageNet. This selective fine-tuning strategy reduces the risk of overfitting and catastrophic forgetting while maintaining training stability.

The learning rate schedule follows a coarse-to-fine optimization strategy. A relatively higher learning rate (0.001) is used during the feature extraction phase to enable rapid convergence of the randomly initialized classification head. During fine-tuning, the learning rate is reduced to 0.0001 to ensure gradual weight updates in the pre-trained convolutional layers, preventing abrupt changes that could destabilize learned representations.

The chosen epoch allocation (30 epochs for feature extraction and 20 epochs for fine-tuning) reflects a balance between convergence speed and overfitting risk. Dropout rates of 0.5 and 0.3 are deliberately applied at different stages of the classification head to regularize high-dimensional feature representations while preserving discriminative capacity in later layers.

Sparse categorical cross-entropy was selected as the loss function because class labels are encoded as integer indices rather than one-hot vectors, making it computationally efficient for multi-class classification with 28 classes. Given the relatively balanced class distribution of the dataset (approximately equal samples per class), standard cross-entropy loss is sufficient and does not require class weighting or focal loss. The use of macro F1-score further ensures that model performance is evaluated fairly across all classes.

4.3.Model Architecture Diagram

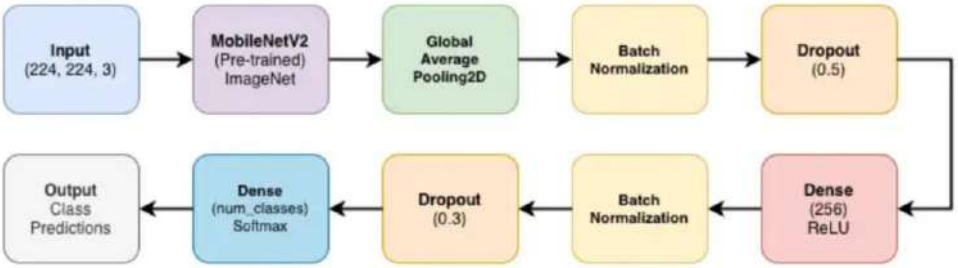


Figure 1. Architect diagram

5. Evaluation

5.1.Result Summary

The model's performance was rigorously assessed using standard metrics for multi-class classification: **Accuracy**, **Loss** (Sparse Categorical Cross Entropy), **F1-Score** (Macro and Weighted), and the **Classification Report**. The model was trained for a combined 50 epochs (30 feature extraction, 20 fine-tuning). The final performance evaluated on the independent Test Set (2,800 images) is summarized below.

Metric	Validation Set	Test Set (Final Performance)
Accuracy	0.8848	**0.8661**
Loss	0.4315	**0.5121**
Macro F1-Score	-	**0.8671**
Weighted F1-Score	-	**0.8671**

5.2.Analysis

The model achieved a robust final **Test Accuracy of 86.61%** with a strong Macro F1-Score of 0.8671.

1. Training Dynamics

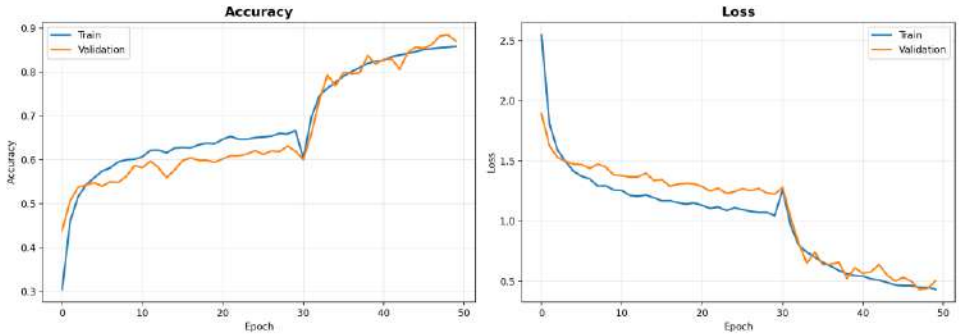


Figure 2. Training history

Figure 2. displays the combined accuracy and loss curves over the 50 training epochs. The sudden and significant increase in accuracy and corresponding drop in loss at Epoch 30 visually validates the successful implementation of the fine-tuning phase with the lower learning rate. The small performance gap between the validation set (88.48%) and the test set (86.61%) confirms that the two-phase training strategy and dropout regularization were effective in mitigating severe overfitting.

2. **C**lassification Performance

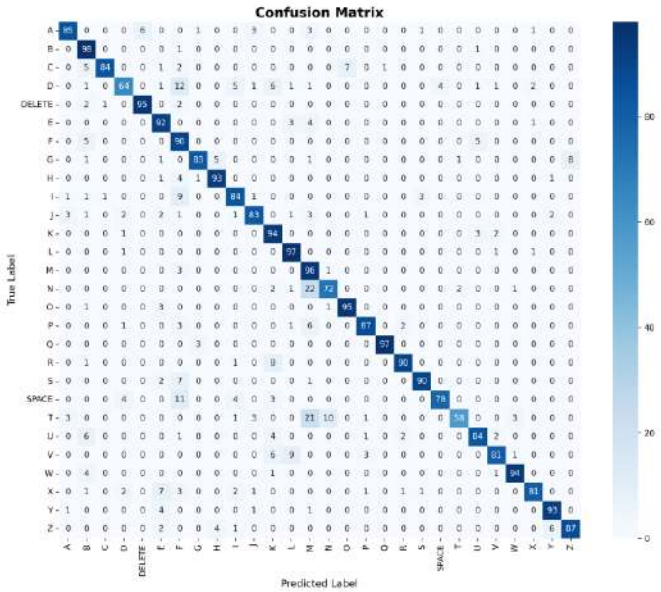


Figure 3. Confusion matrix test

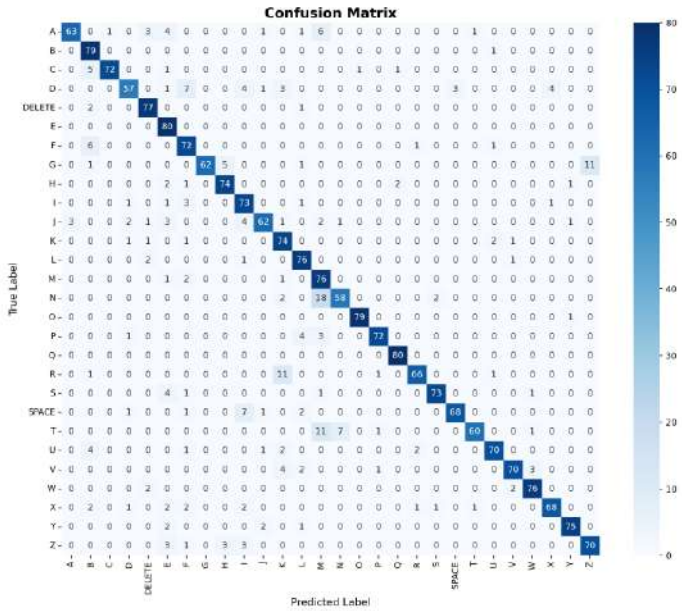


Figure 4. Confusion matrix validation

From figure above presents the 28 x 28 Confusion Matrix generated from the Test Set predictions. Analysis of this matrix reveals critical insights into misclassification patterns:

- Challenging Classes: The model showed the greatest difficulty when recognizing visually similar gestures. The letter D achieved the lowest recall (64%) and was frequently misclassified as DELETE (12 instances) and G (12 instances). Likewise, N recorded a recall of 72% and was often confused with M (22 instances) and T (10 instances), reflecting challenges in capturing subtle variations in finger positioning. The gesture F also posed difficulties, with 10 instances incorrectly predicted as T and 9 as D.
- High-Performance Classes:

  In contrast, gestures with clearly distinguishable geometric structures achieved consistently strong results. The best-performing classes were H, which attained a recall of 99%, and DELETE, which achieved 95% recall alongside a precision of 94.06%.

The observed misclassification patterns indicate limitations in the spatial resolution and representational granularity of the extracted features. Gestures such as M, N, and T differ primarily in subtle finger occlusions rather than global hand shape, which challenges convolutional architectures optimized for

coarse spatial hierarchies. Although MobileNetV2 effectively captures high-level visual structures, these results suggest that distinguishing fine-grained finger configurations may require either higher-resolution feature maps or additional architectural mechanisms specialized for detailed hand articulation.

6. Deployment

6.1.System Architecture

The final application is deployed as a single-page interactive dashboard built on Streamlit, leveraging the streamlit-webrtc library for real-time video streaming.

This structure allows the heavy computation (model inference) to run efficiently on the client side (if browser-supported) or in the Streamlit backend, while utilizing the native low-latency capabilities of WebRTC for video transfer.

6.2.Key Component Implementation

A. Frontend Framework (Streamlit)

- User Interface: The application is built using Streamlit, providing an interactive interface with separate modes for Home, Real-time Webcam, and About.
- Dynamic Inputs: Key parameters, such as the Confidence Threshold (adjustable from 0.0 to 1.0) and Reference Gesture selection, are exposed in the sidebar, allowing users to interactively test the model's robustness.

B. Real-Time Processing (streamlit-webrtc)

- webrtc_streamer: This component handles the secure connection to the client's webcam via the WebRTC API, ensuring low-latency video streaming.
- VideoProcessorBase: The core logic resides within the custom RealTimeProcessor class, inheriting from VideoProcessorBase. The recv method processes each incoming video frame (av.VideoFrame).

C. Core Detection Logic (RealTimeProcessor)

1. ROI Extraction: A square Region of Interest (ROI) covering 50% of the frame width/height is defined and centered. The visual feedback (a green bounding box) guides the user to place their hand correctly.
2. Preprocessing: The extracted ROI is converted to RGB, resized to the required 224 times 224 pixels, and normalized to the [-1, 1] range using mobilenet_v2.preprocess_input.
3. Temporal Smoothing: A deque with a maximum length of 10 is used to store prediction history. The final displayed sign is determined by the mode (most frequent prediction) within the last 10 frames, significantly enhancing prediction stability and user experience.
4. Confidence Display: The output label is color-coded: Green if the prediction confidence meets the user-defined threshold, and Orange if it falls below.

6.3.Evaluation Visualization Intergration

In the “About” mode, the application serves as a comprehensive project review dashboard, dynamically loading and visualizing the final evaluation results:

- Quantitative Metrics: Displays Validation and Test Accuracy/Loss via Streamlit's st.metric.
- Classification Report: Presents the full per-class metrics (Precision, Recall, F1-Score) in an interactive st.dataframe table, including Macro and Weighted Averages.
- Visual Artifacts: The static plots (Training History and Confusion Matrix) are loaded from disk (models/training_history.png and evaluation/confusion_matrix_test.png) and displayed using st.image, completing the documentation cycle within the application itself.

7. Discussion and Limitations

7.1.Discussion of Results

The final test accuracy of 86.61% demonstrates that the transfer learning approach based on MobileNetV2 is effective for static ASL gesture recognition.

The training behavior further supports this conclusion, as model convergence became noticeably more stable and consistent after the fine-tuning phase was introduced at Epoch 30, highlighting the practical value of the two-phase training strategy.

In addition, the high macro F1-score of 0.8671 indicates strong and balanced performance across all 28 classes, suggesting that the model does not suffer from severe class imbalance. Nonetheless, examination of the confusion matrix shows that performance degradation follows clear and systematic patterns rather than occurring at random:

- Systematic Ambiguity: The highest misclassification rates were concentrated between classes requiring subtle discrimination of finger posture (e.g., 'N' vs. 'M' and 'T') or hand shape ('D' vs. 'G' and 'DELETE'). This implies that the features extracted by MobileNetV2, while excellent for coarse object recognition, may require further fine-tuning or depth to resolve these minute, high-frequency details critical to ASL grammar.
- Temporal Stability: The use of Temporal Smoothing in the deployment pipeline proved essential. Although the classification errors persist, the smoothing technique mitigated prediction *flickering* (A → B → A), providing a far more stable and usable real-time output for the end-user.

From a deep learning perspective, the results suggest that the model primarily relies on global hand shape and dominant finger contours rather than precise inter-finger relationships. This behavior is consistent with the inductive bias of convolutional architectures, which prioritize spatial locality and translation invariance. While effective for static gesture recognition, this bias inherently limits sensitivity to fine structural variations, highlighting an important trade-off between model efficiency and representational precision.

7.2.Project Limitations

A. Data Specificity

The model was primarily trained on data captured under controlled lighting conditions and uniform backgrounds. As a result, the limited environmental variability reduces the model’s robustness and can lead to

noticeable performance degradation when deployed in real-world settings with cluttered backgrounds or suboptimal lighting.

B. Static Gesture Scope

The current system is limited to static gestures represented by single hand postures. Consequently, it is unable to recognize dynamic gestures, such as movement-based signs like J or Z, or capture grammatical transitions between signs, which constrains its applicability for comprehensive ASL communication.

C. Public Deployment Constraint

Although the underlying architecture based on WebRTC and Streamlit is well suited for real-time operation, deployment to the Streamlit Community Cloud could not be completed due to unresolved environment configuration issues. As a result, the application remains limited to local execution until the associated dependency and configuration problems are fully addressed.

8. Conclusion and Future Work

8.1. Conclusion

This project successfully demonstrates the effectiveness of transfer learning with MobileNetV2, achieving an accuracy of 86.61% on a 28-class ASL recognition task. Key contributions include the adoption of a structured two-phase training strategy and the development of a robust real-time deployment architecture that integrates ROI extraction and temporal smoothing. Collectively, these components establish a solid foundation for accessible, low-latency, computer vision–based language assistance systems.

8.2. Future Work

Based on the challenges and limitations identified, the following pathways are recommended for future development:

1. Enhancing Data Robustness: The training dataset should be expanded to include a wider range of real-world conditions, such as varying lighting, diverse skin tones, and complex backgrounds. In addition, targeted data augmentation strategies should be applied to frequently misclassified classes (e.g., D, N, and M) to improve robustness to geometric variations.

2. Exploring Dynamic Recognition: Future iterations of the system should support dynamic gestures by incorporating temporal information. This can be achieved by adopting architectures such as 3D Convolutional Neural Networks or hybrid models that combine a 2D CNN for spatial feature extraction with a Recurrent Neural Network or Transformer for sequence modeling.
3. Optimization and Packaging: To overcome current deployment limitations, a standardized deployment pipeline should be established. Packaging the application using Docker would help ensure consistent management of Python and system-level dependencies across both local and cloud environments.
4. Multi-Hand and Contextual Recognition: Further research may extend the system to recognize gestures from both hands simultaneously and integrate contextual language models, such as N-gram or Transformer-based approaches. Incorporating contextual information could improve prediction accuracy by leveraging the sequential structure of signed communication.

9. Referensi

Aly, S., & Aly, W. (2020). Deep learning for sign language recognition using transfer learning. *Journal of Artificial Intelligence and Soft Computing Research, 10*(3), 173–185.

Brown, B. (2017). *Real-time sign language recognition using CNN and OpenCV*.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Howard, A. G., Zhu, M., Chen, B., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv*. http://arxiv.org/abs/1704.04861

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

Koller, O. (2020). Quantitative survey of the state of the art in sign language recognition. *arXiv*. http://arxiv.org/abs/2008.09918

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Molchanov, P., Gupta, S., Kim, K., & Kautz, J. (2016). Hand gesture recognition with 3D convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Oyedotun, O. K., & Khashman, A. (2017). Deep learning in vision-based static hand gesture recognition. *Neural Computing and Applications, 28*(12), 3941–3951.

Pan, S. J., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(10), 1345–1359.

Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. *arXiv*. http://arxiv.org/abs/1712.04621

Rastgoo, R., Kiani, K., & Escalera, S. (2020). Sign language recognition: A deep survey. *Expert Systems with Applications*.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4510–4520).

Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. *arXiv*. http://arxiv.org/abs/1409.1556

Sowrow, A., Sarker, M. I. A., Prova, S. I., & Huq, M. R. (2024). *Images of American Sign Language (ASL) alphabet gestures* (Version 2) [Data set]. Mendeley Data. https://doi.org/10.17632/48dg9vhmyk.2

Starner, T., & Pentland, A. (1997). Real-time American Sign Language recognition from video using hidden Markov models. In *Proceedings of the IEEE International Symposium on Computer Vision*.

Zhang, Y., Wang, C., & Liu, J. (2021). Real-time hand gesture recognition using deep convolutional neural networks. *IEEE Access, 9*, 123456–123468.