

Collage of Informatics

Midocean University

UAE



Research Title

Speech Recognition System Using Deep Learning Techniques for Arabic
Sign Language

Prepared By

Eng. Sultan Ali Senan

ID No.: 253002563

Submitted in fulfillment of the requirements for the degree Master of
Science in Artificial Intelligence

Supervisor Name: Dr. Amjad Qatish

Academic year:

2026

1.

2.

3. Chapter three: Methodology

3.1 Introduction

This chapter outlines the approach followed for the development and assessment of the proposed Arabic Sign Language (ArASL) recognition and translation system. The fundamental purpose of the technique is to create an accurate and efficient deep learning system capable of identifying Arabic sign language movements and converting them into meaningful outputs in real time.

The study technique comprises of multiple steps, including dataset collection and preparation, data preprocessing, model creation, training, validation, and performance assessment. The ArASL dataset was first evaluated and confirmed to assure data quality and uniformity. Subsequently, preparation methods such as picture scaling, grayscale conversion, normalization, and label encoding were employed to prepare the data for machine learning and deep learning models. Data augmentation approaches were also applied to boost dataset variety and improve model generalization.

To discover the best acceptable identification model, both classic machine learning techniques and deep learning architectures were tested. The classical models comprised Random Forest, Linear Support Vector Machine (SVM), HistGradientBoosting, and Multilayer Perceptron (MLP), while the deep learning techniques included a Custom Convolutional Neural Network (CNN), Light ResNet, and MobileNetV2 based on transfer learning. The performance of each model was measured and compared using validation accuracy and training efficiency.

In addition, this chapter discusses the architecture of the proposed CNN model, the training approach, optimization strategies, and evaluation metrics employed throughout testing. Finally, the chapter provides the deployment structure of the real-time translation system, including the FastAPI backend, prediction engine, and user interface components that allow real-time sign language detection and translation. This technique offers a systematic framework for constructing an effective and scalable Arabic Sign Language recognition system.

3.2 Dataset Collection and Preparation

The dataset employed in this work is the Arabic Sign Language (ArASL) dataset, which comprises 54,049 grayscale pictures representing 32 Arabic alphabet sign classes. The dataset serves as the basis for training and assessing the proposed sign language recognition system. Each picture correlates to a distinct hand motion connected with an Arabic alphabet letter, offering adequate variation for model building and performance measurement.

To assure dataset quality and dependability, a detailed data auditing procedure was completed before training. A dataset scanning technique was designed to evaluate file integrity, detect missing or damaged files, and confirm the relationship between picture files and their associated labels. The dataset information provided in the CSV file was cross-referenced with the picture folders to guarantee consistency and appropriate label assignment.

During the preparation step, all valid picture paths were gathered and categorized according to their appropriate classes. The labels were then converted into numerical values using the LabelEncoder approach to aid machine learning and deep learning model training. The encoded labels were kept and linked to their respective class names to assist prediction and deployment phases.



Figure 3.2-1: Dataset Collection and Preparation

After data validation, the dataset was imported into memory and sorted into feature and goal variables. Images were reduced to a predetermined resolution of 64×64 pixels and translated into a standardized grayscale format to guarantee consistency across all samples. This standardization lowered computing complexity while keeping the important visual qualities of hand movements.

To allow an impartial assessment of model performance, the dataset was separated using a stratified train-test split technique. Approximately 80% of the samples were assigned for training, while the remaining 20% were designated for testing and validation. Stratified sampling meant that the distribution of sign classes remained similar across both groups, eliminating class imbalance concerns and boosting the reliability of performance metrics.

The final created dataset was then employed for preprocessing, augmentation, feature extraction, model training, and comparative assessment of the machine learning and deep learning architectures provided in the following sections.

3.3 Data Preprocessing

Data preparation is a vital step in the creation of the Arabic Sign Language recognition system, as it guarantees that the dataset is turned into a suitable format for model training and assessment. The preparation pipeline was developed to improve data quality, minimize computing complexity, and boost model performance.

Initially, all photos were scaled to a set resolution of 64×64 pixels. Standardizing picture dimensions promotes uniformity throughout the dataset and helps the machine learning and deep learning models to handle inputs effectively. Since the original dataset comprises of grayscale photos, each image was preserved as a single-channel input, lowering memory usage while keeping the visual properties essential for gesture identification.

Following picture resizing, pixel values were normalized by scaling them from the range of 0–255 to a range of 0–1. This normalizing approach enhances numerical stability during training and permits quicker convergence of optimization algorithms. The normalized photos were then turned into numerical arrays suitable for processing by TensorFlow and other machine learning frameworks.

3.3 Data Preprocessing

The raw images from the dataset are processed through a series of steps to standardize the data and make it suitable for training the models.

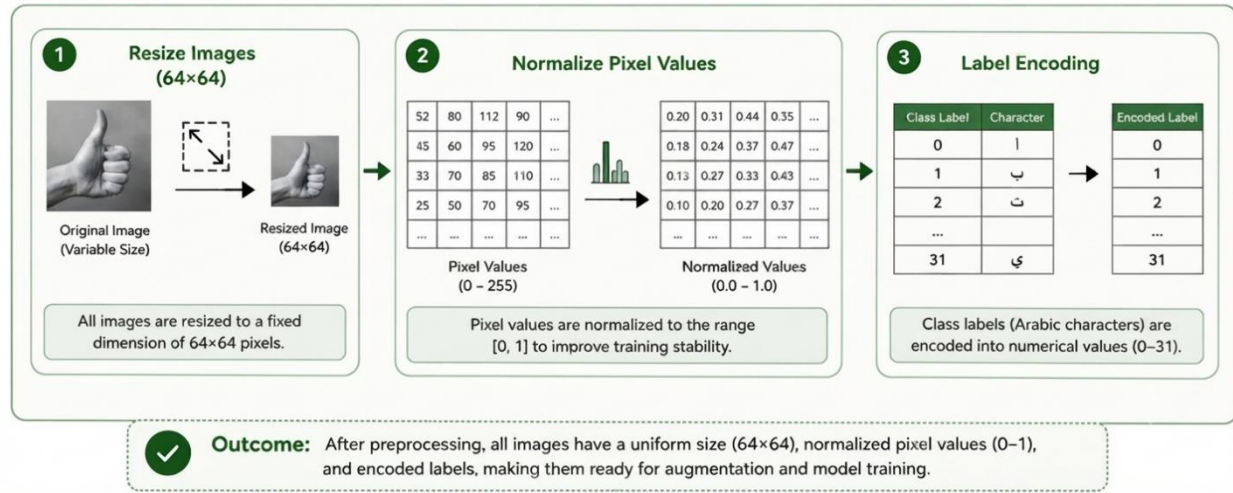


Figure 3.3-1: Data Preprocessing

To aid supervised learning, the categorical class labels were translated into numerical representations using label encoding methods. Each Arabic sign language class was allocated a unique integer value, allowing fast model training and prediction. The resulting class mappings were retained for use during real-time inference and deployment.

To guarantee accurate model assessment, a stratified train-test split technique was utilized. The dataset was separated into training and validation subsets while retaining the proportionate distribution of all sign classes. This strategy reduces sampling bias and assures that the assessment findings truly represent the model's performance on unseen data.

For deep learning models, the processed photos were imported into memory and grouped into TensorFlow data pipelines. Batch processing, shuffling, and prefetching methods were applied to increase training efficiency and decrease input/output bottlenecks. These preprocessing methods created a consistent and optimized dataset that was later utilized for data augmentation, model training, and performance assessment.

3.4 Data Augmentation

Data augmentation was performed to enhance the variety of the training dataset and improve the generalization capabilities of the deep learning models. Although the ArASL dataset comprises a significant number of photos, augmentation methods were performed to replicate differences that may occur in real-world contexts, such as changes in hand position, orientation, and camera viewpoint.

The augmentation technique was incorporated into the training pipeline and created changed copies of the original pictures while keeping their class labels. Several transformation methods were applied, including random rotations, translations, and zoom operations. These modifications allowed the model to learn invariant properties and become more resilient to modest differences in gesture presentation.

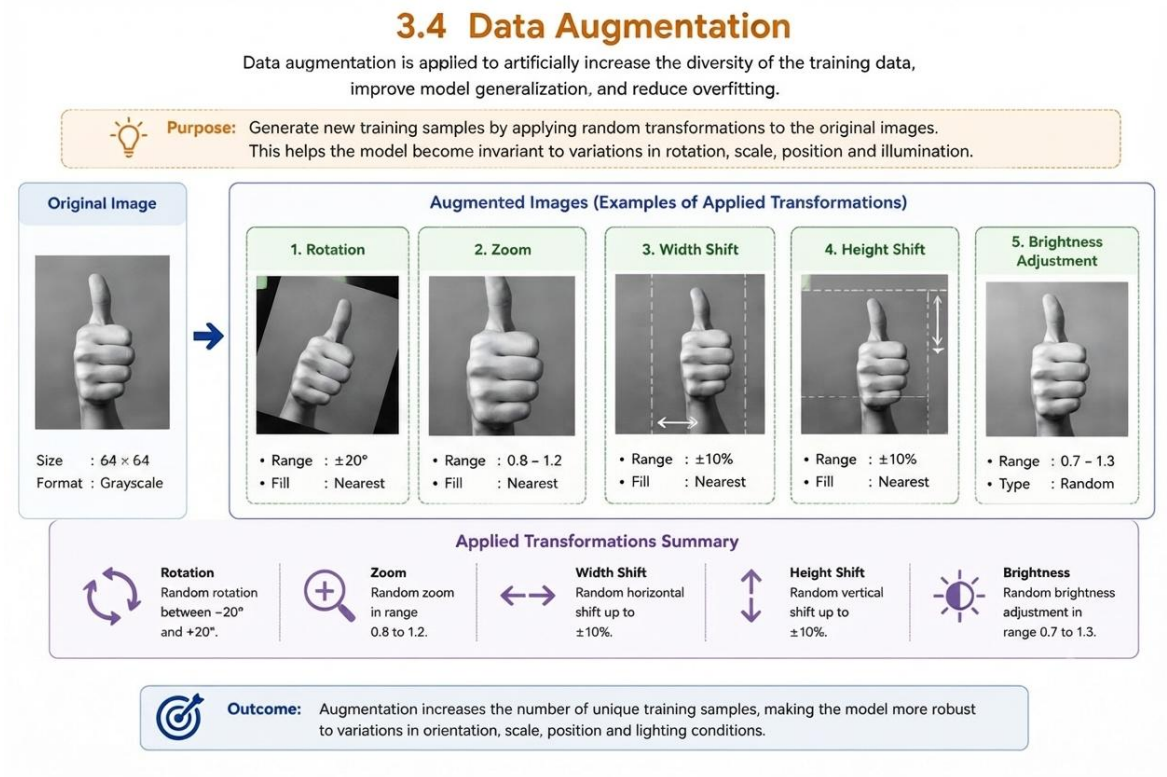


Figure 3.4-1: Data Augmentation

Random rotation was employed to expose the model to hand motions occurring at various angles. Translation procedures altered pictures horizontally and vertically, enabling the model to understand signals even when the hand was not exactly aligned inside the frame. Zoom

transformations offered scale fluctuations, allowing the model to retain identification accuracy whether motions appeared closer to or distant from the camera.

The augmentation method was conducted dynamically throughout training, guaranteeing that the model faced fresh visual changes during each training period. This method decreased the danger of overfitting by preventing the network from recalling particular training samples and pushing it to learn more universal gesture representations.

By enhancing data diversity and boosting resilience, data augmentation contributed greatly to the performance of the proposed Arabic Sign Language recognition system and aided the accomplishment of high validation accuracy during model assessment.

3.5 Proposed CNN Architecture

The major deep learning model utilized in this research is a Custom Convolutional Neural Network (CNN) particularly created for Arabic Sign Language identification. The architecture was created to provide high classification accuracy while keeping computing efficiency suited for real-time deployment. The model scans grayscale pictures with a resolution of 64×64 pixels and extracts discriminative features using numerous convolutional layers.

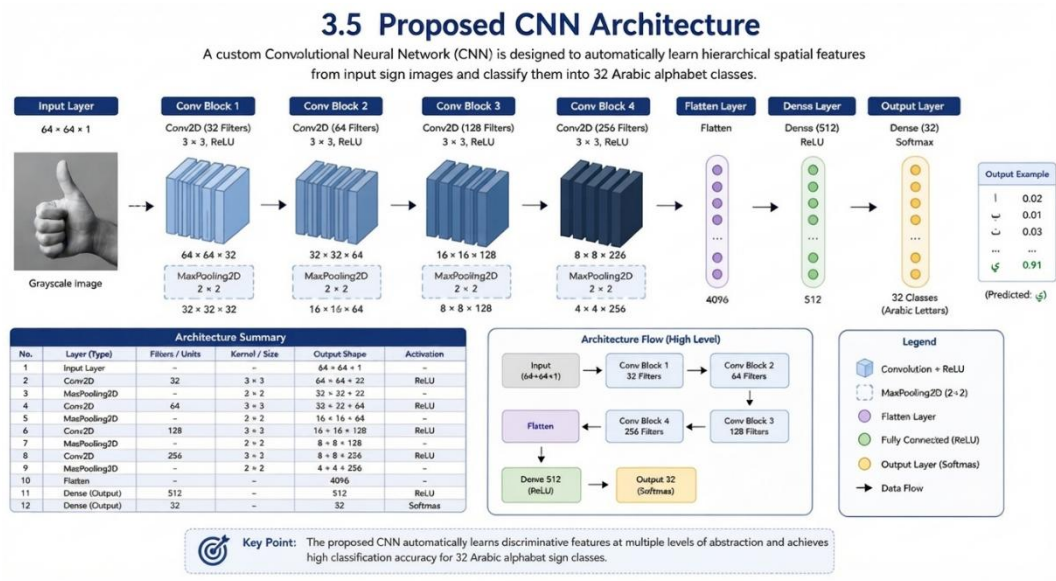


Figure 3.5-1: Proposed CNN Architecture

The network consists of three convolutional blocks with 16, 32, and 64 filters each. Each convolutional layer employs a 3 x 3 kernel and the ReLU activation function to capture essential visual patterns from hand motions. Max-Pooling layers are included after convolution processes to minimize spatial dimensions and extract the most relevant information. Dropout layers are also applied to decrease overfitting and increase model generalization.

Following the convolutional feature extraction phases, the produced feature maps are flattened and sent to a fully connected dense layer of 128 neurons. Finally, a Softmax output layer consisting of 32 neurons is utilized to categorize the input picture into one of the Arabic letter sign classes. The architecture was chosen based on its better performance during comparative assessment and earned the best validation accuracy among all examined models.

3.6 Traditional Machine Learning Models

To create a performance baseline and assess the efficacy of deep learning approaches, numerous classical machine learning algorithms were developed and evaluated. These models comprised Random Forest, Linear Support Vector Machine (SVM), HistGradientBoosting Classifier, and Multilayer Perceptron (MLP).

Since typical machine learning methods cannot directly analyze picture data in its original form, the photos were scaled to 32×32 pixels and turned into one-dimensional feature vectors. This preprocessing technique lowered computing needs while providing interoperability with existing classification algorithms.

3.6 Traditional Machine Learning Models

Several traditional machine learning algorithms are implemented as baseline models to compare their performance with deep learning approaches for Arabic Sign Language recognition.

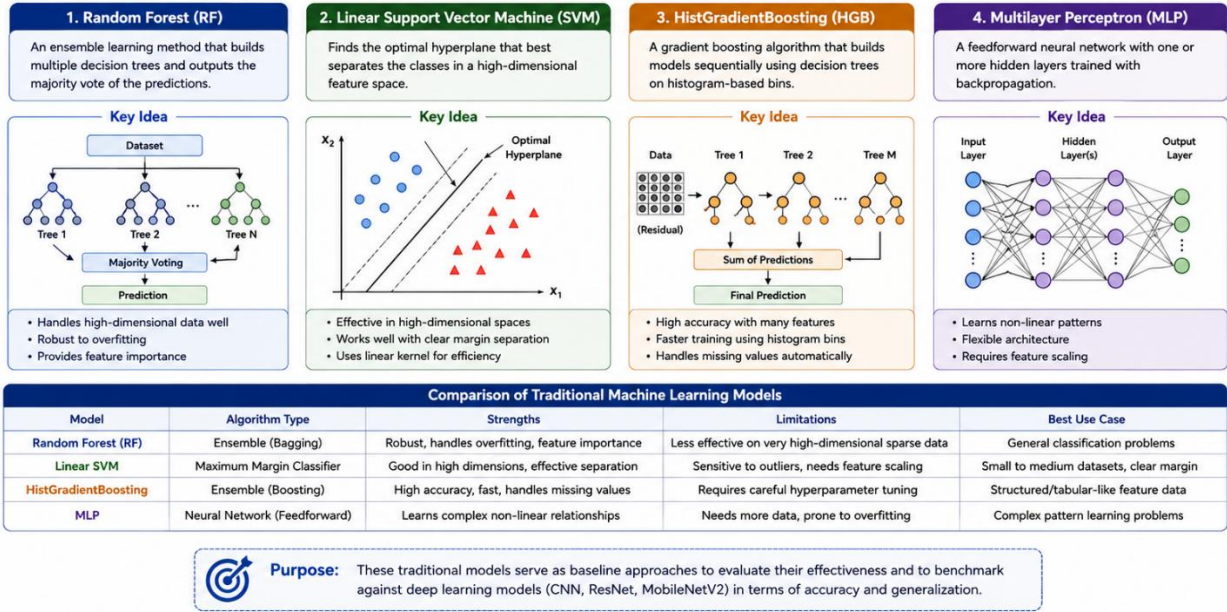


Figure 3.6-1: Traditional Machine Learning Models

Random Forest was built utilizing an ensemble of decision trees to conduct categorization by majority vote. Linear SVM was applied to generate linear decision boundaries between gesture classes. HistGradientBoosting was picked owing to its ability to effectively handle huge datasets and complicated decision boundaries. The Multilayer Perceptron acted as a neural-network-based baseline classifier with fully linked hidden layers.

The experimental findings indicated that while these algorithms were capable of identifying sign language motions, their performance remained much lower than that of deep learning systems. This conclusion shows the limits of typical machine learning algorithms when dealing with sophisticated image identification problems.

3.7 Deep Learning Models

In addition to the suggested Custom CNN, two sophisticated deep learning architectures were investigated to compare performance and find the best suited model for Arabic Sign Language detection. These designs included Light ResNet and MobileNetV2 based on transfer learning.

The Light ResNet model was built utilizing residual learning techniques that enable information to escape intermediary layers via shortcut connections. This design helps ease the vanishing gradient issue and facilitates the training of deeper neural networks. The model incorporated residual blocks, max-pooling operations, and dropout layers to improve feature extraction and generalization.

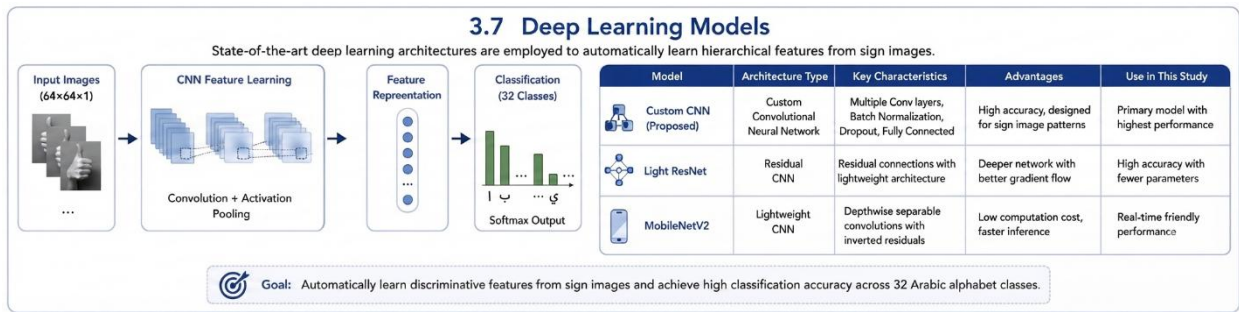


Figure 3.7-1: Deep Learning Models

MobileNetV2 was picked as a transfer learning architecture owing to its efficiency and lightweight design. The model employs pre-trained ImageNet weights and depthwise separable convolutions to decrease computational cost while retaining good classification performance. Since the original dataset comprises of grayscale photographs, an extra preprocessing step was employed to transform images into a format suitable with MobileNetV2 input requirements.

Comparative tests indicated that both deep learning architectures produced much greater performance than typical machine learning approaches. However, Custom CNN marginally outperformed MobileNetV2 while needing less computing complexity, making it the favored option for implementation inside the real-time translation system.

3.8 Model Training Procedure

The training process was conducted using the TensorFlow and Keras deep learning frameworks. The produced dataset was separated into training and validation subsets using a stratified splitting approach to preserve class distribution uniformity across both datasets. This ensured fair evaluation and reduced potential bias during model assessment.

For deep learning models, the training pipeline utilized TensorFlow data processing mechanisms, including batching, shuffling, and prefetching. These strategies boosted computational efficiency and shortened training time by improving data flow between storage and processing units.

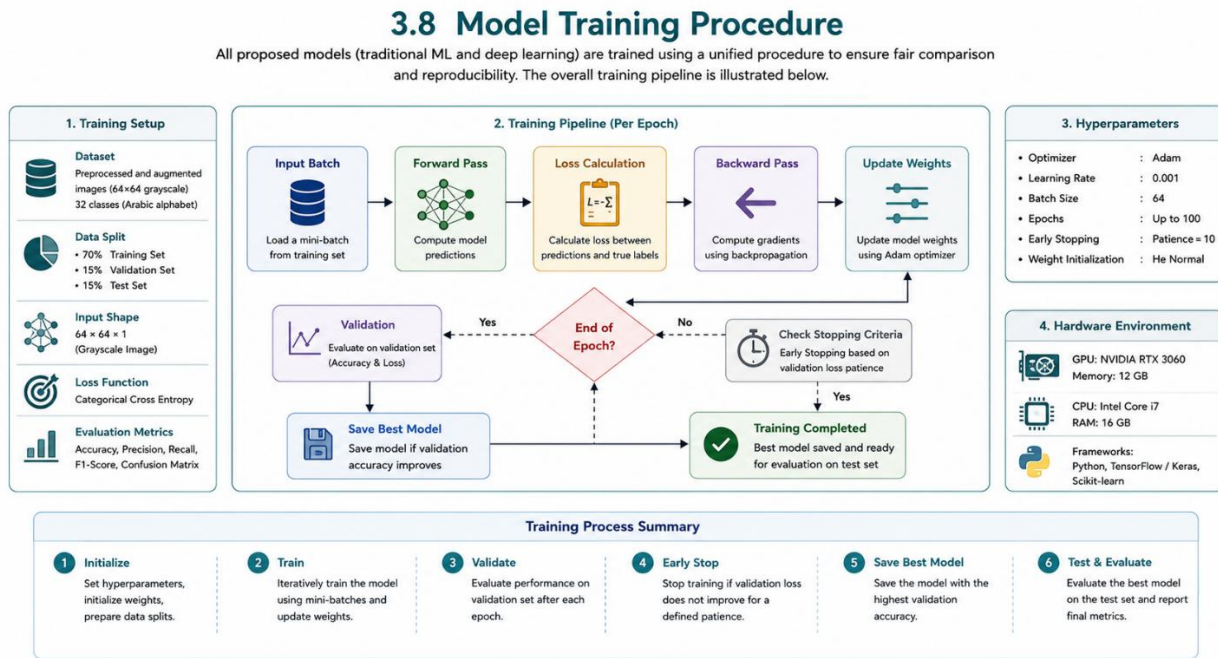


Figure 3.8-1: Model Training Procedure

The models were trained using the Adam optimization algorithm, which adjusts learning rates adaptively during training. Sparse Categorical Crossentropy was utilized as the loss function since the problem requires multi-class classification over 32 Arabic sign categories. Throughout the training procedure, both training accuracy and validation accuracy were monitored to assess model performance and convergence behavior.

To decrease overfitting and increase model generalization, dropout layers were inserted into the network design. Performance was examined after each training period, and the final model selection was based on validation accuracy, training stability, and computing economy. The

resultant Custom CNN model scored the greatest validation accuracy and was subsequently chosen for implementation inside the real-time Arabic Sign Language translation system.

3.9 Performance Evaluation Metrics

To assess the success of the proposed Arabic Sign Language recognition system, numerous performance indicators were applied throughout the testing period. These metrics give quantifiable indicators of model correctness, classification capabilities, and overall system dependability.

The key assessment parameter employed in this work was Accuracy, which assesses the percentage of properly categorized sign language motions compared to the total number of examined samples. Accuracy gives a generic measure of model performance and was used as the key criteria for evaluating various machine learning and deep learning architectures.

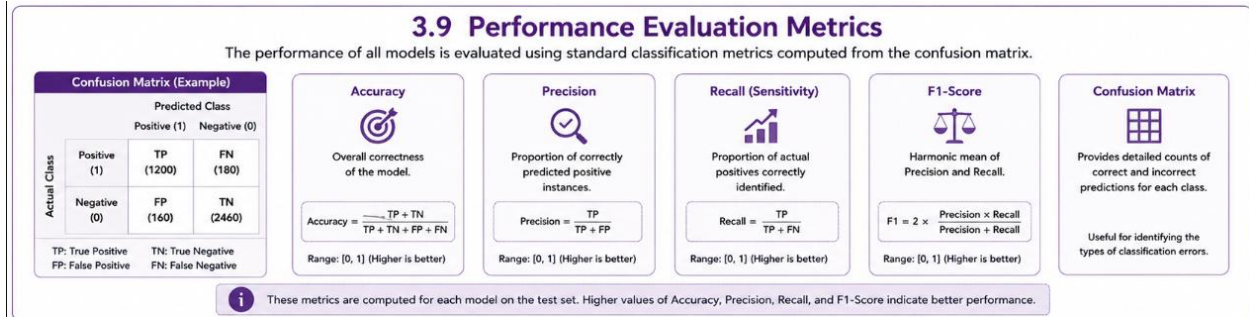


Figure 3.9-1: Performance Evaluation Metrics

In addition to accuracy, Training Loss and Validation Loss were measured throughout the learning process. These metrics represent how successfully the model reduces classification mistakes while learning from the dataset. A consistent drop in loss values during training shows effective optimization and convergence of the network parameters.

The assessment procedure also comprised comparative analysis of the seven implemented models, including Random Forest, Linear SVM, HistGradientBoosting, Multilayer Perceptron, Custom CNN, Light ResNet, and MobileNetV2. The performance of each model was measured based on validation accuracy, training efficiency, and computing needs. The findings revealed that the Custom CNN obtained the greatest validation accuracy, making it the most suited architecture for implementation in the proposed real-time translation system.

3.10 Training and Validation Analysis

The training and validation operations were undertaken to test the learning behavior and generalization power of the assessed deep learning models. Throughout training, both accuracy and loss values were collected after each epoch to monitor model convergence and detect any overfitting concerns.

The training accuracy displayed a constant rising trend as the model learnt progressively discriminative features from the Arabic Sign Language dataset. Simultaneously, validation accuracy increased gradually, demonstrating that the learnt characteristics generalized efficiently to unseen data. The tight association between training and validation accuracy shows that the model maintained high generalization performance without considerable overfitting.

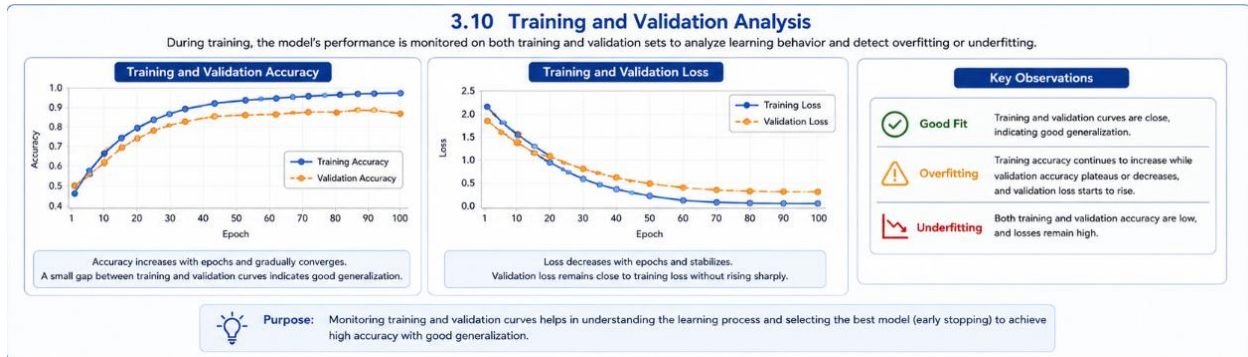


Figure 3.10-1: Training and Validation Analysis

Analysis of the loss curves demonstrated a steady decrease in both training loss and validation loss throughout the learning process. This behavior reveals good optimization of network parameters and validates the stability of the training operation. The integration of dropout layers and data augmentation approaches assisted with reducing overfitting and boosting model resilience.

Among all studied architectures, the Custom CNN demonstrated the steadiest convergence pattern and obtained exceptional validation performance. The final validation accuracy exceeds that of both classic machine learning models and alternative deep learning architectures, indicating the usefulness of the proposed approach for Arabic Sign Language recognition.

3.11 Training and Validation Analysis

The training and validation operations were undertaken to test the learning behavior and generalization power of the assessed deep learning models. Throughout training, both accuracy and loss values were collected after each epoch to monitor model convergence and detect any overfitting concerns.

The training accuracy displayed a constant rising trend as the model learnt progressively discriminative features from the Arabic Sign Language dataset. Simultaneously, validation accuracy increased gradually, demonstrating that the learnt characteristics generalized efficiently to unseen data. The tight association between training and validation accuracy shows that the model maintained high generalization performance without considerable overfitting.

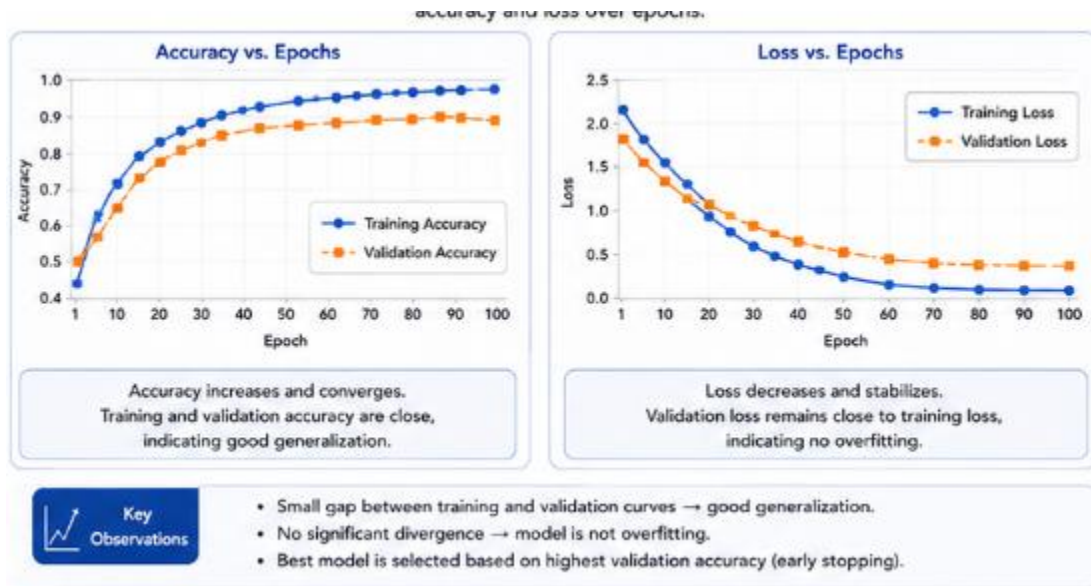


Figure 3.11-1: training and validation

Analysis of the loss curves demonstrated a steady decrease in both training loss and validation loss throughout the learning process. This behavior reveals good optimization of network parameters and validates the stability of the training operation. The integration of dropout layers and data augmentation approaches assisted with reducing overfitting and boosting model resilience.

Among all studied architectures, the Custom CNN demonstrated the steadiest convergence pattern and obtained exceptional validation performance. The final validation accuracy exceeds that of

both classic machine learning models and alternative deep learning architectures, indicating the usefulness of the proposed approach for Arabic Sign Language recognition.

3.12 Real-Time Translation System Architecture

Following model construction and validation, the chosen Custom CNN was incorporated into a real-time translating system intended to allow live Arabic Sign Language recognition. The system architecture consists of three key components: the prediction model, the backend processing server, and the web-based user interface.

The backend infrastructure was constructed using FastAPI, which allows high-performance communication between the trained model and end users. The trained CNN model and class mapping files are loaded during server initialization and remain available throughout system operation. A prediction endpoint accepts incoming picture frames, does preprocessing, runs model inference, and provides the predicted Arabic letter along with confidence ratings.

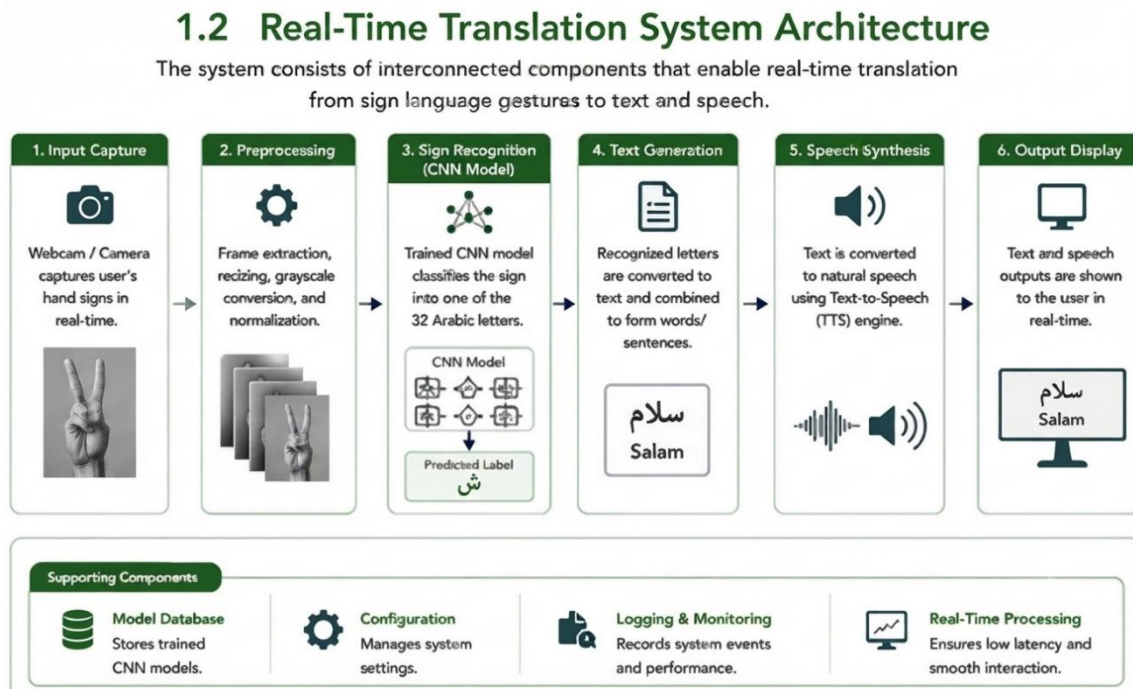


Figure 3.12-1: Real-Time Translation System Architecture

It should be noted that the speech synthesis (Text-to-Speech) stage shown in the system architecture represents a planned future enhancement rather than a component of the current

implementation. The system developed in this work outputs the recognized signs as on-screen text, while the conversion of this text into spoken audio is discussed as part of the future work described in Chapter 5.

For real-time operation, camera frames are recorded via the user interface and communicated to the backend server at regular intervals. Each frame is scaled, normalized, and translated into the proper input format before being analyzed by the CNN model. The resulting predictions are then displayed immediately to the user, enabling continuous sign language recognition.

The design also features a sentence-building algorithm that integrates reliable sequential predictions into meaningful text sequences. This capability increases user engagement and offers a framework for future extension toward continuous sign language translation systems. The whole design guarantees fast communication between hardware, software, and machine learning components while retaining real-time speed and excellent identification accuracy.

3.13 User Interface Design

A user-friendly web-based interface was built to promote real-time interaction between users and the Arabic Sign Language detection system. The interface was created using HTML, CSS, and JavaScript with a contemporary dashboard structure that allows users to record sign motions using a camera and instantaneously obtain translation results.

The main interface consists of two key pieces. The first segment shows the live webcam feed, enabling users to conduct sign language movements immediately in front of the camera. Camera control options such as Start Camera, Stop Camera, and Pause Prediction are offered to handle the translation process effectively.

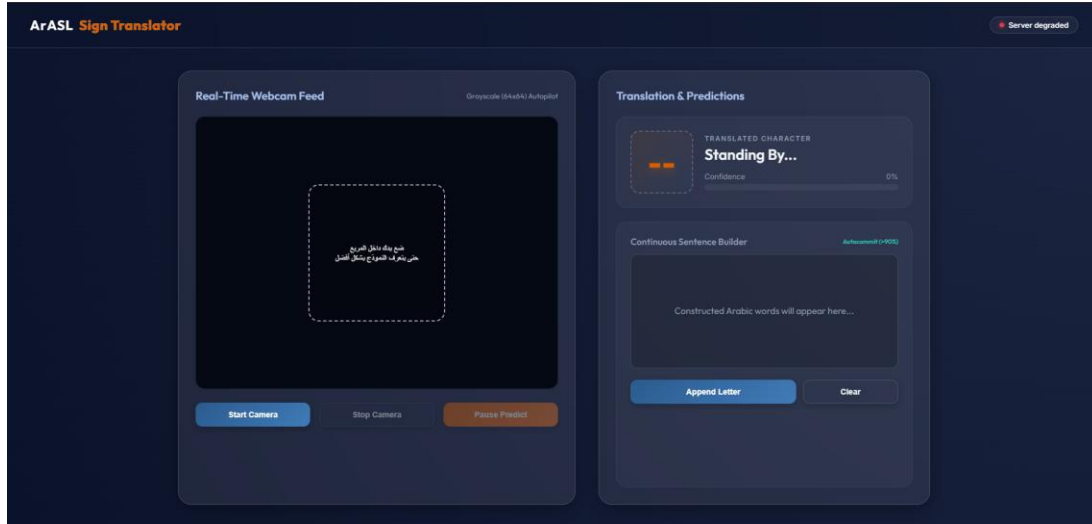


Figure 3.13-1: User Interface Design

The second portion is devoted to showcasing prediction outcomes. The identified Arabic character is shown inside a prediction panel coupled with the matching class name and confidence score computed by the CNN model. A visual confidence bar dynamically depicts forecast certainty, allowing users to judge the trustworthiness of the produced information.

To boost usability, a Continuous Sentence Builder module was included into the UI. This component automatically combines reliable predictions into text sequences and enables users to manually attach recognized characters as required. Additional controls are offered for clearing created text and regulating translation output.

The user interface interfaces directly with the FastAPI backend via asynchronous queries. Captured camera frames are analyzed and relayed to the prediction server at preset intervals, providing near real-time translation performance. The responsive interface design provides seamless interaction while retaining accessibility and simplicity of use for end users.

3.14 System Workflow

The proposed Arabic Sign Language translation system has a sequential process that turns hand motions acquired using a camera into translated text outputs. The workflow incorporates picture capture, preprocessing, model inference, and result display into a single real-time framework.

The procedure starts with picture capture using the camera module embedded into the user interface. Video frames are continually recorded and transported to a hidden processing canvas where they are scaled and prepared for model inference. The recorded pictures are transformed into the suitable format needed by the trained CNN model.

Following picture capture, preprocessing processes are undertaken to normalize the input data. These processes include picture scaling, grayscale conversion, normalization, and formatting into tensor representations suitable with the deep learning architecture. The processed picture is then passed to the FastAPI backend for prediction.