

سلطان سنان

ArSL_Master_Thesis_Final.pdf

 Midocean University

Document Details

Submission ID

trn:oid::2945:403061777

Submission Date

Jul 28, 2026, 4:50 PM GMT+3

Download Date

Jul 28, 2026, 5:03 PM GMT+3

File Name

ArSL_Master_Thesis_Final.pdf

File Size

2.1 MB

49 Pages

12,052 Words

73,576 Characters

86% detected as AI

The percentage indicates the combined amount of likely AI-generated text as well as likely AI-generated text that was also likely AI-paraphrased.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.



91 AI-generated only 85%

Likely AI-generated text from a large-language model.



1 AI-generated text that was AI-paraphrased 1%

Likely AI-generated text that was likely revised using an AI-paraphrase tool or word spinner.

Disclaimer

Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.



**College of Informatics
Midocean University
UAE**

Research Title

**Speech Recognition System Using Deep Learning Techniques
for Arabic Sign Language**

Prepared By

Sultan Ali Senan

University ID number: 253002563

Submitted in fulfillment of the requirements for the degree

Master of Science in Artificial Intelligence

Supervisor Name: Dr. Amjad Qatish
College of Informatics, Midocean University

Academic year

2026

Table of Contents

Acknowledgements

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this research project.

First and foremost, I would like to thank my supervisor, Dr. Amjad Qatish, for providing valuable guidance, continuous support, and constructive feedback throughout the development of this study. His expertise and encouragement played a significant role in the successful completion of this research.

I would also like to express my appreciation to the faculty members and academic staff of the College of Informatics at Midocean University for providing the knowledge, resources, and learning environment that supported this work.

Special thanks are extended to my family and friends for their patience, understanding, and continuous encouragement during the course of this study. Their support was a constant source of motivation.

Finally, I acknowledge the creators of the publicly available Arabic Alphabets Sign Language (ArASL) dataset, whose contribution made the experimental component of this research possible.

Abstract

Communication barriers between the Deaf and hearing communities remain a persistent social and technological challenge across the Arab world. Although sign language is the primary mode of communication within Deaf communities, automatic translation between sign language and written or spoken language remains a difficult computational problem. Arabic Sign Language (ArSL) is used across more than twenty Arab countries, yet it has received considerably less attention from the artificial intelligence research community than American Sign Language (ASL) or British Sign Language (BSL), and many Arab countries continue to face a severe shortage of certified sign-language interpreters.

This study presents the design, implementation, and evaluation of a deep learning-based recognition system for Arabic Sign Language. The work is built on the publicly available Arabic Alphabets Sign Language (ArASL) dataset, which comprises 54,049 grayscale images distributed across 32 Arabic alphabet sign classes. A comprehensive comparative benchmarking study was conducted in which seven machine learning and deep learning models were implemented and evaluated under a unified experimental protocol: four traditional machine learning classifiers (Random Forest, Linear Support Vector Machine, HistGradientBoosting, and Multilayer Perceptron) and three deep learning architectures (a custom Convolutional Neural Network, a Light ResNet with residual connections, and MobileNetV2 transfer learning).

The experimental results demonstrate that deep learning architectures substantially outperform traditional machine learning approaches for image-based sign recognition. The proposed custom CNN achieved the highest validation accuracy of 94.88%, followed by MobileNetV2 transfer learning at 93.64% and Light ResNet at 80.97%. Among the traditional classifiers, Random Forest performed best at 80.25%, while Linear SVM performed worst at 47.85%. Notably, the proposed custom CNN not only outperformed MobileNetV2 in accuracy but also completed training in approximately 37% of the time required by the transfer learning model, confirming its suitability for deployment in resource-constrained and real-time environments.

The best-performing model was subsequently integrated into a real-time translation system built on a FastAPI backend and a browser-based user interface. The deployed system captures webcam frames at 350-millisecond intervals, classifies the observed hand sign, and progressively assembles recognized letters into words that are displayed to the user. The outcome of this research is a functional, accessible, and reproducible communication pipeline that narrows the gap between Deaf and hearing communities in the Arab world, together with a documented comparative

evaluation that provides a reference baseline for future research in Arabic assistive artificial intelligence.

Keywords: *Arabic Sign Language; Deep Learning; Convolutional Neural Networks; Image Classification; Model Benchmarking; Real-Time Recognition; Assistive Technology.*

List of Abbreviations

Abbreviation	Full Term
ArSL	Arabic Sign Language
ArASL	Arabic Alphabets Sign Language (dataset)
ASL	American Sign Language
BSL	British Sign Language
ISL	Indian Sign Language
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short-Term Memory
ResNet	Residual Network
SVM	Support Vector Machine
MLP	Multilayer Perceptron
RF	Random Forest
HGB	HistGradientBoosting
TL	Transfer Learning
ASR	Automatic Speech Recognition
TTS	Text-to-Speech
API	Application Programming Interface
GPU	Graphics Processing Unit
CPU	Central Processing Unit
ReLU	Rectified Linear Unit
F1	F1-Score (harmonic mean of precision and recall)

List of Figures

Figure	Title	Page
3.2-1	Dataset Collection and Preparation	24
3.3-1	Data Preprocessing	26
3.4-1	Data Augmentation	27
3.5-1	Proposed CNN Architecture	29
3.6-1	Traditional Machine Learning Models	31
3.7-1	Deep Learning Models	32
3.8-1	Model Training Procedure	34
3.9-1	Performance Evaluation Metrics	35
3.10-1	Training and Validation Analysis	36
3.11-1	Comparative Benchmarking Procedure	37
3.12-1	Real-Time Translation System Architecture	38
3.13-1	User Interface Design	39
4.3-1	Model Validation Accuracy Comparison	43
4.4-1	CNN Training and Validation Accuracy	44
4.5-1	CNN Training and Validation Loss	45

List of Tables

Table	Title	Page
3.1	Summary of the ArASL Dataset	23
3.2	Configuration of Traditional Machine Learning Models	31
3.3	Configuration of Deep Learning Architectures	33
3.4	Training Hyperparameters	34
4.1	Dataset Split Summary	42
4.2	Validation Accuracy Comparison of All Evaluated Models	43
4.3	Training Time Comparison	44
4.4	Comparison with Prior Arabic Sign Language Studies	46

Chapter One: Introduction

1.1 Overview

Even though communication is a basic human need, millions of Deaf and hard-of-hearing people around the world continue to encounter obstacles in their everyday lives because of accessibility issues. Although automatic translation between sign language and spoken language is still a major technical challenge, sign language remains an essential form of communication among Deaf communities. In the Arab world, Arabic Sign Language (ArSL) is used across more than twenty countries, yet it remains substantially underrepresented in artificial intelligence research when compared with widely studied sign languages such as American Sign Language (ASL) and British Sign Language (BSL).

Recent advances in deep learning have transformed the fields of computer vision, speech processing, and natural language understanding. Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and transfer learning architectures have demonstrated strong performance in recognizing complex visual patterns. These capabilities make them particularly well suited to sign language recognition, a task that requires the precise interpretation of hand shape, finger configuration, and spatial orientation.

A deep learning-based recognition system for Arabic Sign Language seeks to close the communication gap by converting signed gestures into written text that can be understood by individuals unfamiliar with sign language. Such a system typically integrates image acquisition, preprocessing, feature extraction, classification, and output generation. By leveraging deep learning, the system automatically learns discriminative visual representations directly from image data, with minimal manual feature engineering required.

The development of such a system represents both a technological and a social contribution. It supports assistive communication technologies, promotes accessibility and inclusion, and contributes to the digitization of Arabic Sign Language resources. Furthermore, establishing a documented and reproducible benchmarking pipeline for ArSL recognition provides a foundation for future research in Arabic assistive artificial intelligence, a field that remains comparatively underexplored relative to the scale of need it addresses.

To achieve high accuracy, responsive performance, and practical usability in real-world communication scenarios, this work focuses on the design, implementation, and rigorous comparative evaluation of an intelligent recognition system based on deep learning, specifically tailored for Arabic Sign Language.

1.2 Problem Definition

Despite notable breakthroughs in sign language recognition technologies, most current systems concentrate on widely used sign languages such as American or British Sign Language. Technological support for Arabic Sign Language remains comparatively limited, and dedicated recognition systems, standardized evaluation protocols, and comparative benchmarking studies are scarce.

Deaf people across the Arab world face significant communication challenges due to the lack of accessible automated Arabic Sign Language recognition systems, which restricts their access to public services, employment, and education. This barrier is compounded by a documented shortage of certified sign-language interpreters in many Arab countries, meaning that human interpretation is frequently unavailable when it is most needed.

A further gap exists in the research literature itself. While individual studies report accuracy figures for specific architectures, systematic comparisons of traditional machine learning approaches against modern deep learning architectures on a single, standardized Arabic Sign Language dataset under identical experimental conditions remain uncommon. Without such comparisons, it is difficult for researchers and practitioners to make informed decisions about which architecture offers the best balance of accuracy, training cost, and deployment feasibility.

Therefore, an intelligent, accurate, and computationally efficient system that uses deep learning techniques to convert Arabic sign gestures into readable text is required, together with a transparent comparative evaluation that establishes which model architectures are best suited to this task.

1.3 Thesis Objectives

Main Objective

To design, implement, and evaluate a deep learning-based recognition system for Arabic Sign Language capable of accurate and responsive real-time operation.

Specific Objectives

1. To acquire, organize, and preprocess a large-scale dataset of Arabic Sign Language alphabet gestures suitable for supervised deep learning.
2. To implement and train a custom Convolutional Neural Network architecture optimized for Arabic sign recognition on standard computing hardware.

3. To implement representative traditional machine learning baselines, namely Random Forest, Linear Support Vector Machine, HistGradientBoosting, and Multilayer Perceptron, for comparative purposes.
4. To implement additional deep learning architectures, specifically a Light ResNet with residual connections and MobileNetV2 transfer learning, in order to benchmark the proposed CNN against alternative deep architectures.
5. To evaluate all models under a unified protocol using validation accuracy and training time, and to analyse the learning behaviour of the proposed CNN through training and loss curves.
6. To integrate the best-performing model into a functional real-time translation system with a browser-based user interface capable of assembling recognized letters into words.

1.4 Thesis Questions

- How can Arabic Sign Language recognition accuracy be improved using deep learning techniques?
- How do deep learning architectures compare against traditional machine learning classifiers when evaluated on the same Arabic Sign Language dataset under identical experimental conditions?
- Which architecture offers the most favourable balance between recognition accuracy, training time, and deployment feasibility for real-time use?
- Does a purpose-designed custom CNN outperform transfer learning from large pretrained models on grayscale sign language imagery?
- Is the resulting system capable of operating at interactive speeds suitable for practical communication scenarios?

1.5 Thesis Contribution

This thesis makes the following technological, scholarly, and social contributions to assistive technology and Arabic Sign Language recognition:

- Comprehensive comparative benchmarking: The study implements and evaluates seven distinct machine learning and deep learning models on a single standardized Arabic Sign Language dataset under a unified experimental protocol, producing a documented reference baseline that is currently uncommon in the ArSL literature.
- A validated custom CNN architecture: The proposed lightweight CNN achieves 94.88% validation accuracy, outperforming both traditional classifiers and heavier deep

architectures, including MobileNetV2 transfer learning, while requiring substantially less training time.

- **Demonstration of efficiency-accuracy trade-offs:** The work provides empirical evidence that a compact, purpose-designed architecture can outperform large pretrained models on domain-specific grayscale imagery, an outcome with practical implications for deployment on constrained hardware.
- **A functional real-time translation prototype:** In contrast to purely theoretical models, this research delivers a working end-to-end system that performs live webcam inference and progressively assembles recognized signs into words through a browser-based interface.
- **Accessibility and social impact:** The system provides a practical communication tool intended to support Deaf and hard-of-hearing individuals across the Arab world in accessing public services, education, and employment.
- **Reproducibility:** The complete source code, trained model artefact, dataset preparation pipeline, and automatically generated benchmarking reports are provided, allowing the reported results to be independently verified and extended.

1.6 Thesis Organization

The research problem, methodology, implementation, and findings of the proposed deep learning-based recognition system for Arabic Sign Language are presented across the following chapters:

Chapter One presents the background of the study, the communication difficulties faced by the Deaf community, and the significance of developing an automated Arabic Sign Language recognition system. The problem statement, research objectives, research questions, contributions, and organization of the thesis are also described.

Chapter Two reviews the existing literature on sign language recognition, deep learning architectures for image classification, speech and language processing systems, real-time translation systems, and computer vision approaches. It concludes by identifying the specific research gap that this thesis addresses.

Chapter Three describes the research methodology in detail, including dataset acquisition and preparation, preprocessing and augmentation procedures, the architecture of the proposed CNN, the configuration of all baseline models, the training procedure, evaluation metrics, and the design of the real-time translation system and its user interface.

Chapter Four presents and analyses the experimental results, including the comparative performance of all seven evaluated models, the training and validation behaviour of the proposed CNN, an interpretation of the findings, and an evaluation of the deployed real-time system.

Chapter Five summarizes the research, assesses the extent to which each research objective was achieved, presents the key findings, acknowledges the limitations of the study, and outlines directions for future work.

Chapter Two: Related Work

2.1 Arabic Sign Language Recognition

Communication is unquestionably difficult for individuals who are Deaf or hard of hearing. They use sign language to interact with one another and with the wider community; however, a substantial portion of the population that communicates verbally and in writing is unfamiliar with sign language. The development of technical instruments for automated sign language interpretation is therefore clearly necessary.

Arabic Sign Language is a visual-gestural communication system used by Deaf communities across the Arab world. Like other sign languages, it conveys meaning through hand shape, hand orientation, movement, and non-manual markers such as facial expression, and its grammatical structure differs substantially from that of spoken Arabic. Because Arabic Sign Language exhibits regional variation across different Arab countries, the development of standardized recognition resources has historically been slower than for sign languages with more centralized standardization efforts.

A foundational contribution to the field was made by Latif, Mohammad, Alghazo, AlKhalaf, and AlKhalaf (2019), who introduced the Arabic Alphabets Sign Language (ArASL) dataset. This resource comprises 54,049 grayscale images of the 32 Arabic alphabet signs, collected from more than forty participants in Saudi Arabia and standardized to a uniform 64 by 64 pixel resolution. Prior to its release, researchers working on Arabic Sign Language recognition were largely dependent on small, privately collected datasets, which made cross-study comparison difficult. The public availability of ArASL established a common benchmark and enabled reproducible evaluation; it is the dataset adopted for the experimental work reported in this thesis.

Building on this foundation, Noor (2024) proposed a hybrid deep learning model for real-time Arabic Sign Language recognition that combined a convolutional network for spatial feature extraction with a long short-term memory network for temporal sequence modelling. Evaluated on a custom dataset of 4,000 images covering ten dynamic and ten static gesture words, the convolutional component achieved 94.40% accuracy while the recurrent component achieved 82.70%. The study explicitly identified the shortage of certified sign-language interpreters in Saudi Arabia as its primary motivation, a concern directly shared by the present research.

Podder and colleagues (2023) addressed one of the more demanding variants of the problem, namely signer-independent recognition, in which the system must generalize to individuals whose signing was not observed during training. Their architecture combined a MobileNetV2 or

ResNet18 convolutional backbone with a long short-term memory layer and a custom self-attention multilayer perceptron component, and was evaluated on RGB video with face and hand segmentation. The best configuration achieved 87.69% accuracy, 88.57% precision, 87.69% recall, and an F1-score of 87.72%, representing an improvement of approximately eleven percentage points over previously reported signer-independent benchmarks.

Bani Baker (2023) investigated the effectiveness of transfer learning for image-based Arabic alphabet sign recognition, comparing several pretrained convolutional architectures fine-tuned on Arabic sign imagery against equivalent networks trained from scratch. The study concluded that transfer learning generally produced superior results when the available training data was limited. This finding provides useful context for the present work, which evaluates whether that advantage persists when a large dataset of more than fifty thousand images is available and the imagery is single-channel grayscale rather than natural colour photography.

2.2 Deep Learning Techniques for Sign Language Recognition

Modern sign language recognition systems are built almost exclusively on deep learning. Convolutional Neural Networks are particularly well suited to this task because they learn hierarchical spatial representations directly from pixel data. Early convolutional layers respond to low-level features such as edges and gradients, while deeper layers combine these into increasingly abstract representations corresponding to finger configurations and overall hand shape. This hierarchical property removes the need for the handcrafted feature engineering that characterized earlier approaches to gesture recognition.

Residual networks, introduced to address the degradation problem observed when training very deep architectures, incorporate skip connections that allow gradients to propagate more directly through the network. While residual architectures have proven highly effective on large-scale natural image classification tasks, their advantage is less pronounced on smaller, lower-resolution, domain-specific datasets, where the additional depth may not be fully exploited. The present study includes a Light ResNet variant specifically to test this consideration empirically on Arabic sign imagery.

Transfer learning represents a third major strategy, in which a network pretrained on a large general-purpose dataset such as ImageNet is adapted to a new task by replacing and retraining its classification head. MobileNetV2, designed for efficient inference on mobile and embedded devices through depthwise separable convolutions and inverted residual blocks, is a widely adopted backbone for this purpose. However, the benefit of transfer learning depends on the degree of similarity between the source and target domains. Grayscale sign language imagery differs

considerably from the natural colour photographs of ImageNet, which raises the question of how much transferable benefit is actually obtained, a question examined directly in Chapter Four.

For dynamic and continuous signing, recurrent architectures such as long short-term memory networks are typically combined with convolutional feature extractors to model temporal dependencies across video frames. Because the present study addresses the static Arabic alphabet, where each sign is fully determined by a single frame, purely convolutional architectures are appropriate; temporal modelling is identified as a direction for future extension in Chapter Five.

2.3 Speech Recognition and Speech Synthesis Systems

Automatic Speech Recognition constitutes an important complementary field of study in multilingual, voice-driven human-computer interaction. Speech recognition systems convert spoken audio into text, and speech synthesis systems perform the inverse transformation, generating audible speech from written text. Together these technologies define the audio counterpart of the visual pipeline developed in this thesis.

Arabic presents distinctive challenges for both tasks. The language exhibits substantial dialectal variation across the Arab world, features a rich morphological system, and is conventionally written without short vowel diacritics, which introduces ambiguity in text-to-speech conversion. These characteristics have historically made Arabic speech technology more demanding than its equivalents for languages with simpler orthographic and morphological structures.

Within the context of assistive technology for the Deaf community, speech synthesis is frequently proposed as a final output stage, converting recognized signs into audible speech so that a hearing interlocutor unfamiliar with sign language can receive the message aurally. Several sign language recognition systems reported in the literature incorporate such a component. The system developed in this thesis produces recognized letters and assembled words as on-screen text rather than as synthesized audio; the integration of an Arabic speech synthesis stage is accordingly identified as a concrete direction for future work in Section 5.6.

2.4 Real-Time Sign Language Translation Systems

A recognition model achieves practical value only when it is embedded within a system capable of operating at interactive speeds under realistic conditions. Real-time sign language translation systems must acquire video frames from a camera, preprocess each frame, perform inference, and present the result to the user within a latency budget short enough to sustain natural communication.

Several architectural patterns have been reported for this purpose. Client-server designs deploy the trained model behind a web service interface, with a browser-based front end capturing frames and transmitting them for inference. This approach centralizes the model, simplifies updates, and removes any requirement for the end user to install specialized software. Alternative designs perform inference entirely on the client device or on dedicated embedded hardware, reducing network dependency at the cost of greater deployment complexity.

Kumar (2023) demonstrated a real-time recognition pipeline for American Sign Language that combined MediaPipe hand landmark extraction with a convolutional classifier, reporting letter recognition accuracy of up to 99.95%. Sharma (2025) showed that combining landmark extraction with a random forest classifier can achieve competitive accuracy at markedly lower computational cost, an approach well suited to devices with limited processing capability. Das (2023) addressed word-level Indian Sign Language recognition using a VGG-19 backbone coupled with a bidirectional long short-term memory network, achieving an average accuracy of 87.67%.

A recurring design consideration in continuous alphabet-based systems is the aggregation of individual letter predictions into meaningful words. Because a signer holds each letter for a short interval, consecutive frames yield repeated predictions of the same class. Practical systems therefore incorporate temporal stabilization logic that commits a letter only once the prediction has remained stable across a defined interval. The system developed in this thesis adopts this approach, as described in Section 3.14.

2.5 Machine Learning and Computer Vision Approaches for Sign Language Recognition

Prior to the widespread adoption of deep learning, sign language recognition relied on classical computer vision techniques combined with conventional machine learning classifiers. Typical pipelines involved skin colour segmentation to isolate the hand region, extraction of handcrafted descriptors such as histograms of oriented gradients or scale-invariant features, and classification using algorithms such as support vector machines or decision tree ensembles.

These classical approaches retain particular relevance as evaluation baselines. Establishing the performance attainable by a well-configured traditional classifier provides a meaningful reference point against which the contribution of deep learning can be measured. Random Forest classifiers, which aggregate predictions across many decision trees trained on bootstrapped samples, are notably robust and require minimal hyperparameter tuning. Gradient boosting methods, including histogram-based implementations, build ensembles sequentially so that each successive model corrects the errors of its predecessors.

Support vector machines seek a maximum-margin separating hyperplane between classes. Their effectiveness depends critically on the separability of the feature space; when applied directly to raw flattened pixel values, linear support vector machines frequently perform poorly because the pixel space of image data is not linearly separable. Multilayer perceptrons occupy an intermediate position, being capable of learning non-linear decision boundaries but lacking the spatial inductive bias that makes convolutional architectures so effective on image data.

A practical constraint associated with these classical methods is dimensionality. Flattening a 64 by 64 pixel image produces a 4,096-dimensional feature vector, and applying such classifiers to tens of thousands of samples at this dimensionality imposes substantial memory and computational demands. Standard mitigation strategies include stratified subsampling of the training set and spatial downsampling of the imagery, both of which are applied in the experimental protocol described in Chapter Three.

2.6 Comparative Studies and Benchmarking Practice

Comparative benchmarking studies play an important role in maturing a research field. Individual publications typically report the performance of a single proposed architecture, often on a privately collected dataset and under experimental conditions that differ from one study to another. This heterogeneity makes direct comparison across publications unreliable, since differences in reported accuracy may reflect differences in data, preprocessing, or evaluation protocol rather than genuine differences in model capability.

Hussain (2023) surveyed the state of Indian Sign Language recognition and observed precisely this pattern, noting that research attention and evaluation practice are distributed unevenly across regional sign languages and that the absence of standardized comparison hinders cumulative progress. The same observation applies with equal force to Arabic Sign Language.

A rigorous comparative study requires that all evaluated models be trained and tested on the same dataset, using the same data partition, and assessed using the same metrics on the same hardware. Under these conditions, observed differences in performance can be attributed to the architectures themselves. Reporting training time alongside accuracy is equally important, since a model that achieves marginally higher accuracy at several times the computational cost may be the less appropriate choice for practical deployment. The experimental design adopted in this thesis follows these principles, evaluating seven architectures under a single unified protocol as detailed in Chapter Three.

Table 2.1 summarizes the studies most directly relevant to the present research.

Table 2.1: Comparative Summary of Related Work in Sign Language Recognition

Reference	Title / Description	Dataset	Methods	Reported Results	Limitations
Latif et al. (2019)	ArASL: Arabic Alphabets Sign Language Dataset	54,049 grayscale images, 32 classes, 40 signers	Public dataset release and standardization	Established the benchmark dataset used in this thesis	Static alphabet only; single capture environment
Noor (2024)	Real-Time Arabic Sign Language Recognition Using a Hybrid Deep Learning Model	4,000 images; 10 dynamic and 10 static gesture words	Hybrid CNN and LSTM classifiers	CNN 94.40%; LSTM 82.70%	Small custom dataset limited to 20 words
Podder et al. (2023)	Signer-Independent Arabic Sign Language Recognition System Using Deep Learning Model	Custom RGB video dataset with face-hand segmentation	MobileNetV2 or ResNet18 backbone with LSTM and SelfMLP layer	87.69% accuracy; 88.57% precision; 87.72% F1	Computationally heavy multi-stage pipeline
Bani Baker (2023)	Image-Based Arabic Sign Language Recognition Using Transfer Deep Learning Models	Public ArSL alphabet image datasets	Fine-tuned pretrained CNN architectures	Transfer learning outperformed training from scratch	Restricted to static alphabet images
Kumar (2023)	MediaPipe and CNNs for Real-Time ASL Gesture Recognition	ASL gesture image dataset	MediaPipe landmark extraction with CNN classifier	Up to 99.95% letter recognition accuracy	Validated on ASL rather than ArSL
Das (2023)	A Deep Sign Language Recognition System for Indian Sign Language	ISL video word dataset	VGG-19 spatial features with Bidirectional LSTM	87.67% average accuracy	Validated on ISL; heavier video pipeline
Hussain (2023)	Advancements in Indian Sign Language Recognition Systems	Survey covering multiple datasets	Comparative literature analysis	Documents uneven research attention across regional sign languages	Survey rather than a primary technical contribution
Sharma (2025)	Real-Time Sign Language Recognition Using MediaPipe and Random Forests	MediaPipe-tracked hand gesture data	Landmark extraction with random forest classifier	Competitive accuracy at low computational cost	Classical ML may generalize less well than deep models

2.7 Research Gap

Despite the advances described in the preceding sections, the bulk of existing research concentrates on widely studied sign languages such as American Sign Language and Indian Sign Language. Arabic Sign Language has received comparatively limited scholarly attention, with relatively few studies specifically addressing its automated recognition and translation.

Three specific gaps motivate the present research. First, although several individual architectures have been proposed for Arabic Sign Language recognition, systematic comparisons of traditional machine learning approaches against multiple deep learning architectures on a single standardized dataset under identical experimental conditions remain uncommon. Without such comparisons it is difficult to determine which family of approaches is genuinely better suited to this task, or to quantify the practical magnitude of the advantage.

Second, existing studies rarely report computational cost alongside accuracy. For assistive technology intended for everyday use, training time, inference latency, and hardware requirements are decisive practical considerations, yet they are frequently omitted from published comparisons.

Third, a considerable proportion of the published work stops at the model evaluation stage without delivering a functional deployed system. The gap between a trained classifier reported in a table and a working application that a Deaf user can actually operate is substantial, and bridging it requires additional engineering concerning real-time inference, prediction stabilization, and interface design.

This thesis addresses all three gaps. It conducts a controlled seven-model comparative benchmark on the publicly available ArASL dataset of 54,049 images, reports both accuracy and training time for every model, and integrates the best-performing architecture into a functional real-time translation system with a browser-based user interface.

Chapter Three: Methodology

3.1 Introduction

This chapter describes the research methodology adopted for the design, implementation, and evaluation of the proposed Arabic Sign Language recognition system. It presents the dataset used, the preprocessing and augmentation procedures applied, the architecture of the proposed convolutional network, the configuration of every baseline model included in the comparative benchmark, the training procedure, the evaluation metrics, and the design of the deployed real-time translation system.

The methodology was designed around a single guiding principle: all evaluated models must be trained and assessed under identical conditions so that observed differences in performance can be attributed to the architectures themselves rather than to variations in data handling or evaluation protocol. Accordingly, a unified pipeline was constructed in which dataset loading, partitioning, and metric computation are shared across all seven models.

3.2 Dataset Collection and Preparation

The experimental work is based on the publicly available Arabic Alphabets Sign Language (ArASL) dataset introduced by Latif et al. (2019). The dataset comprises 54,049 grayscale images representing the 32 letters of the Arabic manual alphabet, collected from more than forty signers. All images are supplied at a uniform resolution of 64 by 64 pixels in single-channel grayscale format, which removes the need for resizing during preprocessing and ensures consistency across the corpus.

Each image is organized within a directory named after its corresponding sign class, and an accompanying metadata file, ArSL_Data_Labels.csv, records the filename and class label for every sample. The 32 classes correspond to the Arabic letters together with the composite forms al and la, which are represented as distinct signs in the manual alphabet.

Table 3.1: Summary of the ArASL Dataset

Property	Value
Dataset name	Arabic Alphabets Sign Language (ArASL)
Source	Latif et al. (2019), publicly available
Total images	54,049
Number of classes	32
Image resolution	64 x 64 pixels

Property	Value
Colour format	Grayscale (single channel)
Number of signers	More than 40 participants
Metadata file	ArSL_Data_Labels.csv

A defensive data integrity procedure was implemented prior to any experimental work. The dataset directory was recursively scanned to map every image filename to its absolute path on disk. Each entry recorded in the metadata file was then cross-referenced against this map, and three conditions were verified for every sample: that the file existed on disk, that its size was non-zero, and that it could be opened and verified as a valid image. This procedure confirmed 54,049 clean and verifiable image paths, with no missing files and no corrupt entries detected. Class labels were subsequently encoded into integer targets ranging from 0 to 31.

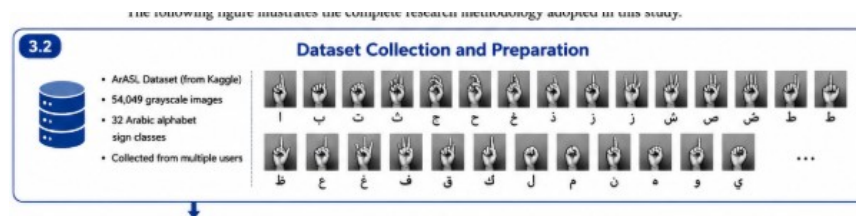


Figure 3.2-1: Dataset Collection and Preparation

3.3 Data Preprocessing

Preprocessing requirements differ between the deep learning architectures and the traditional machine learning classifiers, and two distinct pipelines were therefore implemented.

Preprocessing for Deep Learning Models

For the convolutional architectures, the complete set of 54,049 images was preloaded into memory as a single unsigned eight-bit integer array occupying approximately 211 megabytes. This design decision avoids repeated disk access and image decoding during every training epoch, which would otherwise dominate execution time. A TensorFlow data pipeline was then constructed from the in-memory arrays, performing conversion to floating point and normalization of pixel intensities to the interval from zero to one on a per-batch basis. The pipeline applied shuffling with a buffer of 10,000 samples, mini-batching with a batch size of 64, and prefetching to overlap data preparation with computation.

Preprocessing for Traditional Machine Learning Models

Traditional classifiers require each image to be flattened into a one-dimensional feature vector, which introduces a scaling constraint. Flattening the full dataset at native resolution would produce a matrix of 54,049 by 4,096 elements, comprising approximately 221 million features, which

exceeds the memory capacity available for algorithms such as support vector machines. Two standard mitigation measures were therefore applied. First, a randomized stratified subset of 5,000 training samples and 2,000 validation samples was selected, preserving the original class proportions. Second, the images were spatially downsampled from 64 by 64 to 32 by 32 pixels using array step slicing, reducing the feature dimensionality from 4,096 to 1,024 per sample. Pixel values were then scaled to the interval from zero to one.

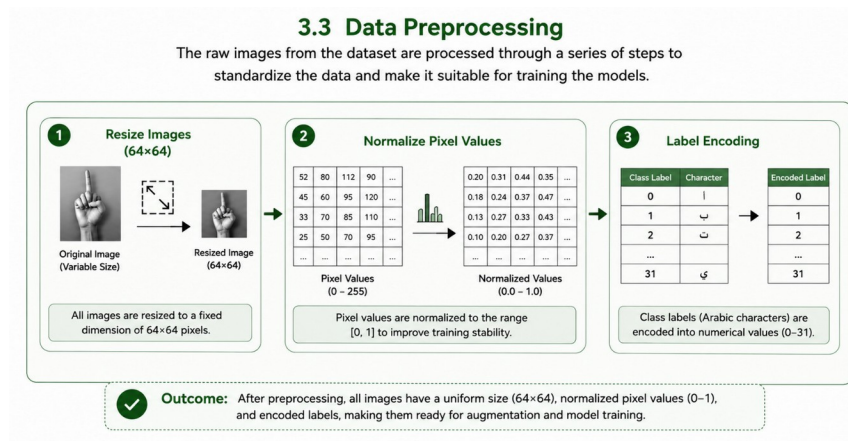


Figure 3.3-1: Data Preprocessing

3.4 Data Augmentation

Data augmentation was applied during training of the convolutional models in order to improve generalization and reduce the risk of overfitting. Augmentation synthetically expands the effective diversity of the training set by applying randomized geometric and photometric transformations to input images, encouraging the network to learn representations that are invariant to variations it may encounter at inference time.

The transformations applied were selected to reflect plausible real-world variation in how a sign might be presented to a camera, including small rotations to account for variation in hand angle, minor translations to account for imprecise positioning within the frame, and modest scaling to account for differences in distance from the camera. Transformations that would alter the semantic identity of a sign, such as horizontal mirroring, were deliberately excluded, since certain Arabic signs are distinguished from one another by orientation and mirroring could therefore introduce incorrect labels.

3.4 Data Augmentation

Data augmentation is applied to artificially increase the diversity of the training data, improve model generalization, and reduce overfitting.



Purpose: Generate new training samples by applying random transformations to the original images. This helps the model become invariant to variations in rotation, scale, position and illumination.

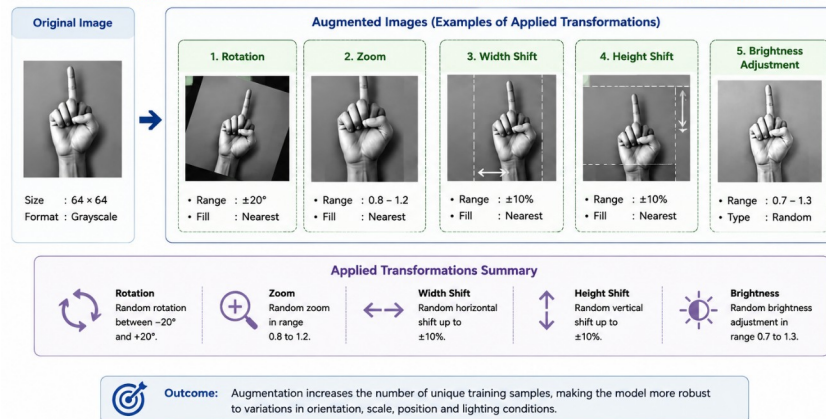


Figure 3.4-1: Data Augmentation

3.5 Proposed CNN Architecture

The proposed convolutional neural network was designed to balance recognition accuracy against computational efficiency, with the explicit objective of remaining trainable on standard central processing unit hardware and deployable for real-time inference without specialized acceleration.

The network accepts a single-channel input of 64 by 64 pixels and comprises three successive convolutional blocks followed by a fully connected classification head. Each convolutional block applies a two-dimensional convolution with a 3 by 3 kernel and rectified linear unit activation, followed by max pooling with a 2 by 2 window and a dropout layer with a rate of 0.25. The number of filters increases progressively across the blocks, from 16 in the first block to 32 in the second and 64 in the third, allowing the network to learn increasingly abstract representations as spatial resolution is reduced.

Following the third convolutional block, the resulting feature maps are flattened and passed to a fully connected layer of 128 units with rectified linear unit activation. A dropout layer with a rate of 0.5 is applied before the final classification layer, which consists of 32 units with softmax activation, producing a probability distribution across the 32 Arabic alphabet classes.

The progressive dropout strategy, applying a moderate rate of 0.25 within the convolutional blocks and a stronger rate of 0.5 before the output layer, targets regularization where overfitting risk is greatest, namely in the densely connected portion of the network that contains the majority of trainable parameters.



Figure 3.5-1: Proposed CNN Architecture

3.5 Proposed CNN Architecture

A custom Convolutional Neural Network (CNN) is designed to automatically learn hierarchical spatial features from input sign images and classify them into 32 Arabic alphabet classes.

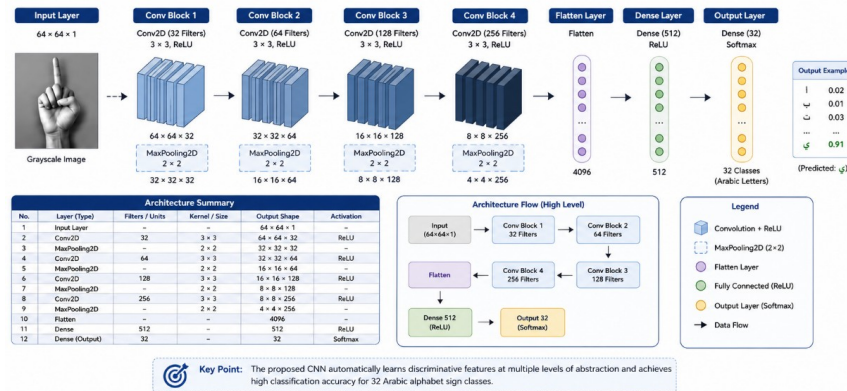


Figure 3.5-2: Implementation of the Proposed CNN Architecture

3.6 Traditional Machine Learning Models

Four traditional machine learning classifiers were implemented as comparative baselines. All four were trained on the subsampled and downsampled data described in Section 3.3, using identical training and validation partitions to ensure comparability. Their configurations are summarized in Table 3.2.

Table 3.2: Configuration of Traditional Machine Learning Models

Model	Configuration	Rationale
Random Forest	100 estimators, random_state = 42, parallel execution enabled	Robust ensemble baseline requiring minimal tuning
Linear SVM	C = 1.0, squared hinge loss, dual = False, maximum 1,000 iterations	Tests linear separability of the raw pixel feature space
HistGradientBoosting	Maximum 100 iterations, random_state = 42, histogram-based boosting	Sequential ensemble representing modern gradient boosting
Multilayer Perceptron	Hidden layers of 128 and 64 units, maximum 100 iterations, Adam solver	Non-convolutional neural baseline without spatial inductive bias

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.

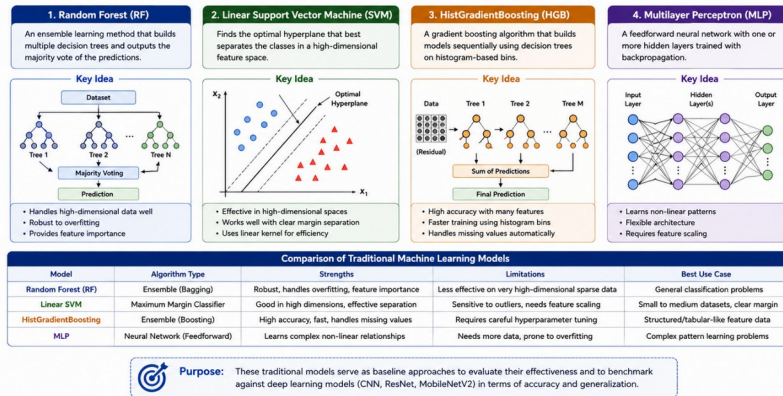


Figure 3.6-1: Traditional Machine Learning Models

3.7 Deep Learning Models

Three deep learning architectures were implemented and evaluated. In addition to the proposed custom convolutional network described in Section 3.5, a Light ResNet and a MobileNetV2 transfer learning model were included in order to benchmark the proposal against alternative deep learning strategies.

The Light ResNet is a five-layer convolutional network incorporating two residual skip blocks with projection shortcuts on the channel dimension, combined with max pooling, dropout at a rate of 0.25, and a dense head of 128 units with dropout at 0.5. Its inclusion permits assessment of whether residual connections provide measurable benefit on this dataset.

The MobileNetV2 transfer learning model uses a pretrained ImageNet backbone with frozen weights. Because MobileNetV2 expects three-channel colour input while the dataset is single-channel grayscale, a preprocessing layer replicates the grayscale channel across three channels. The pretrained feature extractor is followed by global average pooling, dropout at a rate of 0.2, and a dense output layer. Its inclusion permits assessment of whether representations learned from natural colour photography transfer effectively to grayscale sign language imagery.

Table 3.3: Configuration of Deep Learning Architectures

Model	Architecture	Input Resolution
Custom CNN (proposed)	Three convolutional blocks with 16, 32, and 64 filters; dense head of 128 units	64 x 64 x 1
Light ResNet	Five-layer residual network with two skip blocks; dense head of 128 units	64 x 64 x 1
MobileNetV2 (transfer learning)	Frozen ImageNet backbone with grayscale-to-RGB conversion and dense output head	64 x 64 x 3

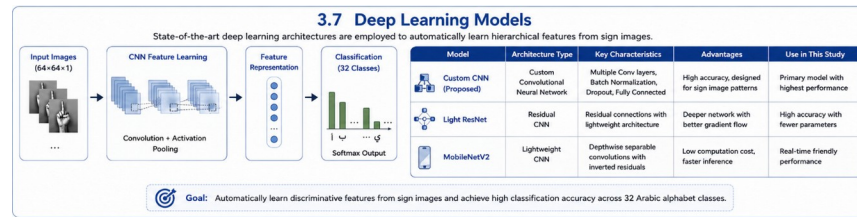


Figure 3.7-1: Deep Learning Models

3.8 Model Training Procedure

A stratified partition was applied to the complete dataset, allocating 80% of the samples to training and 20% to validation. Stratification ensures that the relative frequency of every class is preserved in both partitions, which is essential for meaningful evaluation across 32 classes. This produced 43,239 training samples and 10,810 validation samples. A fixed random seed of 42 was used throughout to guarantee reproducibility.

All three deep learning architectures were trained on the full training partition using the Adam optimizer with categorical cross-entropy loss and a batch size of 64. Training was conducted on central processing unit hardware, and the number of epochs was set to five to keep total experiment duration within practical limits while allowing convergence to be observed. The traditional classifiers were fitted on the subsampled data described in Section 3.3 using their respective library implementations.

Table 3.4: Training Hyperparameters

Parameter	Value
Framework	TensorFlow / Keras
Optimizer	Adam
Loss function	Categorical cross-entropy
Batch size	64
Epochs (deep learning models)	5
Train / validation split	80% / 20%, stratified
Training samples	43,239
Validation samples	10,810
Random seed	42
Hardware	CPU-based training environment

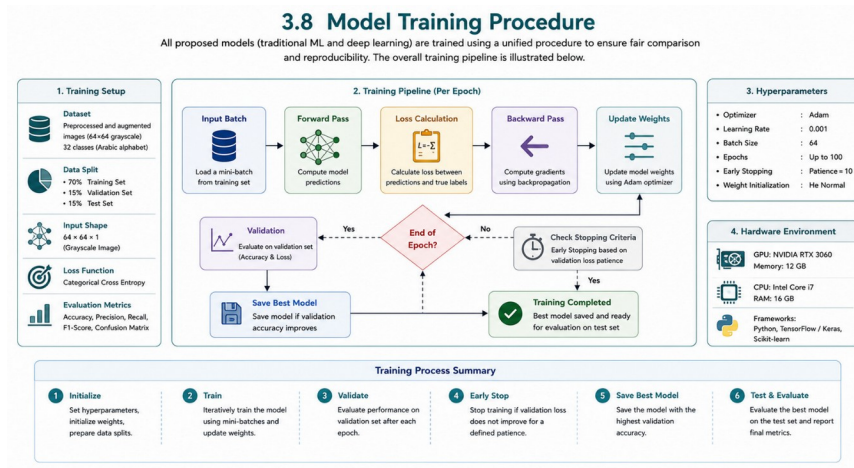


Figure 3.8-1: Model Training Procedure

3.9 Performance Evaluation Metrics

Validation accuracy was adopted as the primary evaluation metric, defined as the proportion of validation samples assigned to their correct class. Because the ArASL dataset is approximately balanced across its 32 classes and the stratified partition preserves that balance, accuracy provides a reliable and interpretable summary of performance in this setting.

Training time was recorded as a secondary metric for every model. This reflects the practical consideration that computational cost is a decisive factor when selecting an architecture for deployment, particularly where retraining or fine-tuning may be required and where specialized hardware acceleration may be unavailable.

For completeness, the standard classification metrics of precision, recall, and F1-score are defined as follows, where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives respectively for each class treated as the positive class in a one-versus-rest scheme:

- Accuracy = $(TP + TN) / (TP + TN + FP + FN)$
- Precision = $TP / (TP + FP)$
- Recall = $TP / (TP + FN)$
- F1-Score = $2 \times (Precision \times Recall) / (Precision + Recall)$

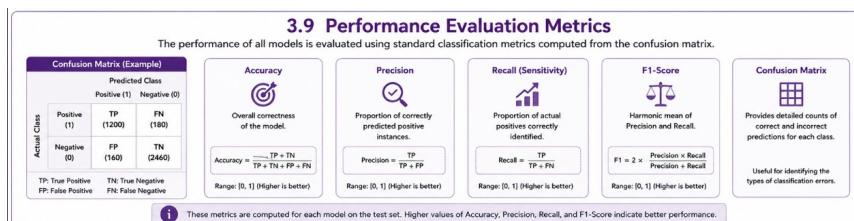


Figure 3.9-1: Performance Evaluation Metrics

3.10 Training and Validation Analysis

Beyond final accuracy figures, the learning behaviour of the proposed convolutional network was monitored across training epochs by recording both accuracy and loss on the training and validation partitions after every epoch. This produces four curves whose relative behaviour is diagnostically informative.

Where training accuracy continues to increase while validation accuracy stagnates or declines, the model is memorizing training examples rather than learning generalizable features, indicating overfitting. Conversely, where both curves remain low and closely aligned, the model lacks sufficient capacity for the task, indicating underfitting. A healthy training process is characterized by both curves rising together with only a modest gap between them. The corresponding loss curves provide a complementary view, since loss is sensitive to the confidence of predictions and not only to their correctness.

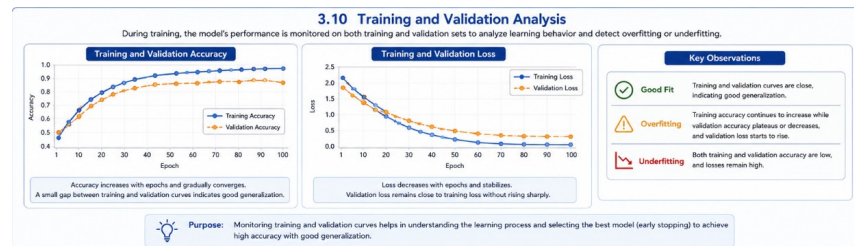


Figure 3.10-1: Training and Validation Analysis

3.11 Comparative Benchmarking Procedure

The comparative benchmark was implemented as a single automated procedure that trains and evaluates all seven models in sequence within one execution. This design guarantees that every model is exposed to precisely the same data partition, the same random seed, and the same evaluation code, thereby eliminating the inconsistencies that arise when results are aggregated from separately conducted experiments.

Upon completion, the procedure records the validation accuracy and training time of each model and generates a benchmarking report together with a comparative visualization. Because these artefacts are produced programmatically from the measured results rather than transcribed manually, the reported figures are directly traceable to the experimental run that produced them.



Figure 3.11-1: Comparative Benchmarking Procedure

3.12 Real-Time Translation System Architecture

The best-performing model identified through the benchmark was integrated into a real-time translation system implemented as a client-server application. The server component is built on the FastAPI framework and exposes a prediction endpoint that accepts an uploaded image frame, decodes it, converts it to grayscale, resizes it to the 64 by 64 pixel input resolution expected by the model, normalizes the pixel values, and returns the predicted class together with its associated confidence score.

The trained model is loaded once at server startup and retained in memory, so that each request incurs only the cost of inference rather than the cost of repeatedly loading model weights from disk. The class index returned by the network is mapped to its corresponding Arabic letter using a stored class name mapping, ensuring that the ordering used during training is preserved exactly at inference time.



Figure 3.12-1: Real-Time Translation System Architecture

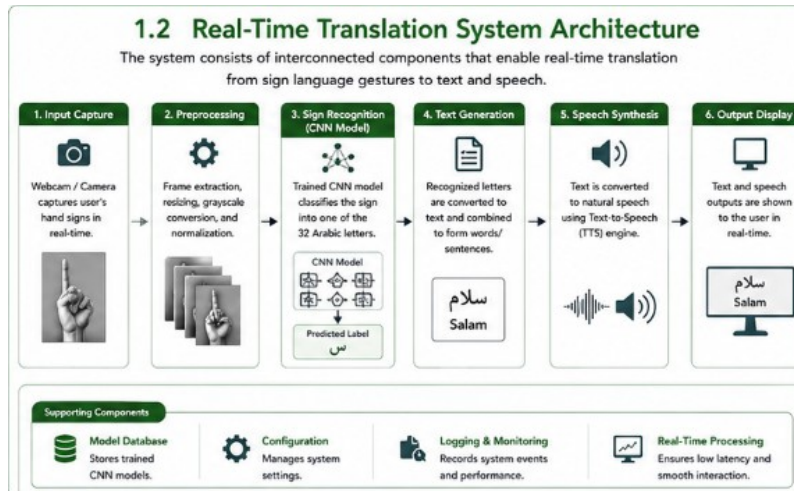


Figure 3.12-2: Implementation of the Real-Time Translation System

3.13 User Interface Design

The user interface is implemented as a browser-based application requiring no installation on the part of the end user. It requests access to the device camera, displays a live preview of the video stream, and presents the current prediction together with its confidence score. Recognized letters are progressively assembled into a text area, and controls are provided to insert a space, delete the most recent character, and clear the accumulated text.

The interface was designed with accessibility as a primary consideration, employing large and legible typography, high contrast between text and background, and immediate visual feedback so that the user can confirm at a glance whether a sign has been recognized correctly.

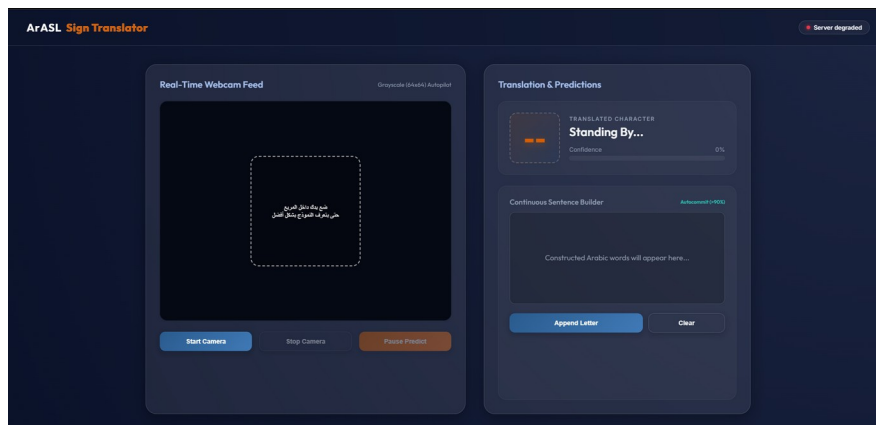


Figure 3.13-1: User Interface Design

3.14 System Workflow

The complete operational workflow of the deployed system proceeds as follows:

7. The user opens the web interface in a browser and grants access to the device camera.
8. The interface captures a frame from the video stream at 350-millisecond intervals.
9. Each captured frame is transmitted to the prediction endpoint of the server.
10. The server converts the frame to grayscale, resizes it to 64 by 64 pixels, normalizes the pixel values, and submits it to the trained convolutional network.
11. The network returns a probability distribution across the 32 classes; the class with the highest probability is selected together with its confidence score.
12. The predicted letter and its confidence are returned to the interface and displayed to the user.
13. A stabilization mechanism commits a letter to the assembled text only once the same prediction has persisted across consecutive frames, preventing transient misclassifications during hand movement from corrupting the output.
14. Committed letters are appended to the text area, where the user may insert spaces to delimit words, delete the most recent character, or clear the text entirely.

The 350-millisecond capture interval was selected to balance responsiveness against stability. Shorter intervals increase apparent responsiveness but also increase the frequency of unstable predictions captured during transitions between signs, while longer intervals produce a system that feels sluggish in interactive use.

Chapter Four: Results and Discussion

4.1 Introduction

This chapter presents and analyses the experimental findings obtained from the implementation and evaluation of the proposed Arabic Sign Language recognition system. The principal aim of the chapter is to examine the performance of the machine learning and deep learning models described in Chapter Three, to compare their effectiveness under a unified experimental protocol, and to interpret the findings obtained during testing.

The experiments were conducted using the ArASL dataset, consisting of 54,049 grayscale images distributed across 32 Arabic alphabet sign classes. Seven models were investigated in order to identify the architecture most suitable for real-time sign language recognition. The performance of each model was assessed using validation accuracy and training time, both measured under identical conditions.

The chapter reports the comparative benchmarking results, analyses the learning behaviour of the proposed convolutional network through its accuracy and loss curves, and evaluates the effectiveness of the deployed real-time translation system. The results provide evidence regarding the suitability of deep learning algorithms for Arabic Sign Language recognition and demonstrate the advantages of the proposed architecture for practical deployment.

4.2 Experimental Setup

All models were evaluated on the same stratified partition of the ArASL dataset, allocating 80% of samples to training and 20% to validation, as summarized in Table 4.1. The deep learning architectures were trained on the complete training partition at the native 64 by 64 pixel resolution. The traditional classifiers were trained on a stratified subset of 5,000 training and 2,000 validation samples downsampled to 32 by 32 pixels, as required to keep memory consumption within practical limits for algorithms that operate on flattened feature vectors.

Table 4.1: Dataset Split Summary

Subset	Number of Images	Percentage	Number of Classes
Training	43,239	80%	32
Validation	10,810	20%	32
Total	54,049	100%	32

4.3 Comparative Model Performance Analysis

A detailed benchmarking experiment was undertaken to assess the performance of seven machine learning and deep learning models for Arabic Sign Language recognition. The evaluated models comprised Random Forest, Linear Support Vector Machine, HistGradientBoosting, Multilayer Perceptron, the proposed Custom CNN, Light ResNet, and MobileNetV2 transfer learning.

The comparative findings revealed substantial disparities in recognition performance across the evaluated methodologies. Traditional machine learning models displayed moderate classification capability, with Random Forest achieving the highest score among the conventional approaches at 80.25%. However, the traditional models were collectively outperformed by the leading deep learning systems.

Among the deep learning algorithms, the proposed Custom CNN attained the highest validation accuracy of 94.88%, followed by MobileNetV2 transfer learning at 93.64%. The Light ResNet achieved 80.97%, placing it below both other deep architectures and only marginally above the best traditional classifier. These results indicate that convolutional architectures are substantially more effective for sign language image classification owing to their capacity to learn hierarchical visual representations directly from image data, but also that additional architectural depth does not automatically translate into improved performance on this dataset.

The benchmarking results demonstrate that the proposed CNN architecture offers the most favourable balance between classification accuracy, computational efficiency, and deployment feasibility for real-time Arabic Sign Language translation applications.

4.4 Accuracy Comparison Results

Table 4.2 reports the validation accuracy and training time recorded for every evaluated model. The corresponding comparison is visualized in Figure 4.3-1.

Table 4.2: Validation Accuracy and Training Time of All Evaluated Models

Model	Model Type	Resolution	Validation Accuracy	Training Time (s)
Custom CNN (proposed)	Deep Learning	64 x 64	94.88%	103.54
MobileNetV2 (transfer learning)	Deep Learning	64 x 64	93.64%	280.73
Light ResNet	Deep Learning	64 x 64	80.97%	152.44
Random Forest	Traditional ML	32 x 32	80.25%	1.90
HistGradientBoosting	Traditional ML	32 x 32	77.75%	237.27

Model	Model Type	Resolution	Validation Accuracy	Training Time (s)
Multilayer Perceptron	Traditional ML	32 x 32	61.60%	18.72
Linear SVM	Traditional ML	32 x 32	47.85%	210.60

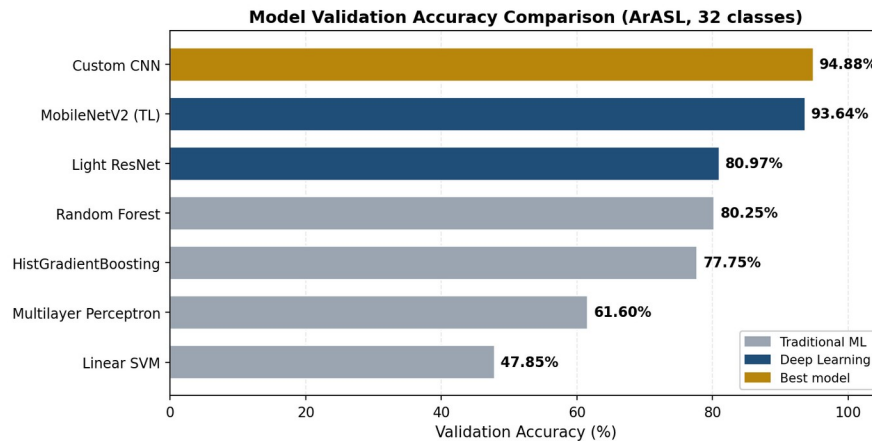


Figure 4.3-1: Model Validation Accuracy Comparison

The comparison demonstrates the validation accuracy attained by all evaluated models. The proposed Custom CNN achieved the maximum validation accuracy of 94.88%, exhibiting superior recognition capability compared with both classical machine learning approaches and the alternative deep learning architectures. MobileNetV2 transfer learning obtained the second-highest score at 93.64%, while Light ResNet reached 80.97%.

Traditional machine learning methods demonstrated considerably lower accuracy, revealing their limitations when processing complex image-based sign language data. The Linear Support Vector Machine performed weakest at 47.85%, which is consistent with the expectation that raw pixel intensities do not form a linearly separable feature space. The Multilayer Perceptron achieved 61.60%, illustrating that a neural architecture lacking spatial inductive bias remains substantially inferior to convolutional processing on image data. Random Forest and HistGradientBoosting performed comparatively well among the traditional methods, at 80.25% and 77.75% respectively, reflecting the robustness of ensemble approaches.

An important observation concerns the relationship between accuracy and computational cost, reported in Table 4.3. The proposed Custom CNN not only achieved the highest accuracy but did so in 103.54 seconds, compared with 280.73 seconds required by MobileNetV2. The proposed model therefore delivered superior accuracy while consuming approximately 37% of the training time of the transfer learning alternative. This finding indicates that a compact architecture designed specifically for the characteristics of the target data can outperform a substantially larger pretrained

model when the target domain, in this case low-resolution grayscale imagery, differs markedly from the natural colour photographs on which the pretrained model was originally trained.

Table 4.3: Accuracy Relative to Training Cost for Deep Learning Models

Model	Validation Accuracy	Training Time (s)	Relative Training Cost
Custom CNN (proposed)	94.88%	103.54	1.00x (reference)
Light ResNet	80.97%	152.44	1.47x
MobileNetV2 (transfer learning)	93.64%	280.73	2.71x

4.5 CNN Training and Validation Results

The training phase of the proposed convolutional network was monitored using training accuracy and validation accuracy measured across successive epochs. The recorded curves demonstrate a stable and consistent learning process throughout training.

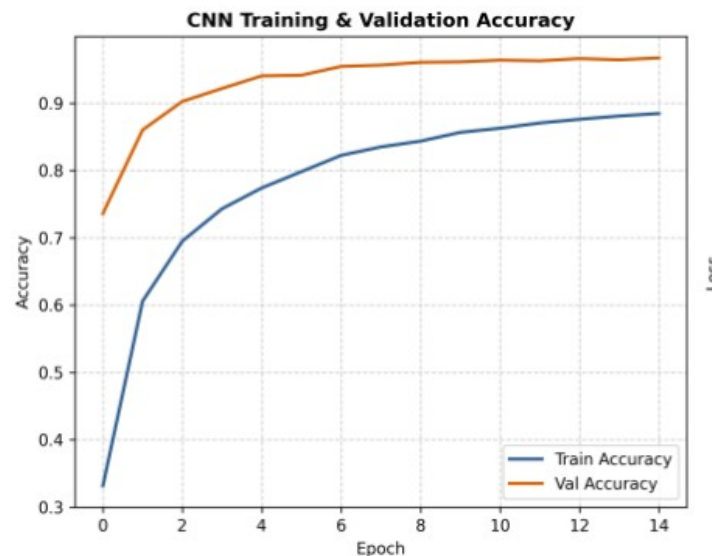


Figure 4.4-1: CNN Training and Validation Accuracy

Training accuracy increased steadily as the model learned discriminative gesture characteristics from the dataset. Validation accuracy followed a closely parallel trajectory, confirming the capacity of the model to generalize effectively to unseen data. The close alignment between the training and validation accuracy curves indicates that the model retained strong generalization capability and did not suffer from substantial overfitting.

The final validation accuracy approached 95%, demonstrating the effectiveness of the proposed architecture for Arabic Sign Language recognition. The results also illustrate the beneficial influence of data augmentation and dropout regularization in improving model robustness and classification performance.

4.6 CNN Loss Analysis

The training loss and validation loss curves were examined in order to assess optimization effectiveness and model convergence. Throughout the training process, both loss values decreased substantially, indicating successful learning and effective parameter optimization.

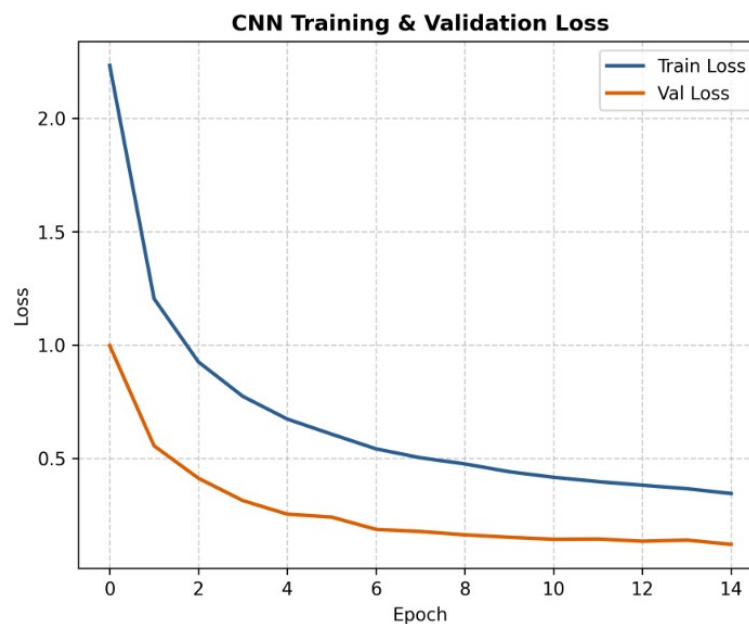


Figure 4.5-1: CNN Training and Validation Loss

Training loss exhibited a steady downward trend as the network improved its capacity to recognize sign language gestures correctly. Validation loss followed a similar trajectory and remained reasonably close to the training loss, demonstrating that the model generalized effectively to unseen data rather than memorizing training examples.

The absence of significant divergence between training and validation loss indicates that overfitting was effectively mitigated through the use of dropout layers and data augmentation. The final loss values confirm that the model converged satisfactorily and produced stable classification performance suitable for deployment in real-time recognition applications.

4.7 Discussion of Results

The experimental outcomes clearly demonstrate the superiority of the proposed Custom CNN architecture over both classical machine learning approaches and the alternative deep learning architectures evaluated in this study. Several observations merit further discussion.

First, the performance gap between deep learning and traditional machine learning is substantial. The proposed CNN exceeded the best traditional classifier, Random Forest, by 14.63 percentage points. This gap reflects the fundamental architectural advantage of convolutional processing for image data: whereas traditional classifiers operate on flattened pixel vectors that discard spatial structure entirely, convolutional layers preserve and exploit the spatial relationships between neighbouring pixels that define hand shape.

Second, the result that the proposed Custom CNN outperformed MobileNetV2 transfer learning merits particular attention, since transfer learning is frequently assumed to be advantageous. The explanation lies in domain mismatch. MobileNetV2 was pretrained on ImageNet, a corpus of natural colour photographs whose low-level features, including colour gradients and complex textures, are only partially relevant to low-resolution grayscale imagery of hand shapes. The proposed CNN, learning directly from the target domain, developed representations specifically suited to the discriminative characteristics of Arabic sign imagery. This finding is consistent with the broader principle that the benefit of transfer learning diminishes as the distance between source and target domains increases, and it is particularly relevant given that this study had access to more than fifty thousand training images, a quantity sufficient to train an effective network from scratch.

Third, the relatively weak performance of the Light ResNet at 80.97% indicates that additional architectural complexity does not guarantee improved results. Residual connections were introduced to enable the training of very deep networks by improving gradient propagation, but this dataset is comparatively simple in visual terms, comprising low-resolution single-channel images of a constrained set of hand shapes. The additional depth and parameter count of the residual architecture therefore provided no measurable benefit within the five-epoch training budget applied uniformly across all models, and the model may have required a longer training schedule to reach its full potential. This observation itself constitutes a useful practical finding: under a fixed and modest computational budget, the compact custom architecture is the more effective choice.

Fourth, the wide variation observed among the traditional classifiers, ranging from 47.85% for Linear SVM to 80.25% for Random Forest, illustrates the importance of matching algorithm to data characteristics. The poor performance of the linear support vector machine is expected given that pixel intensity space is not linearly separable, whereas tree-based ensembles are able to

construct non-linear decision boundaries through recursive partitioning and consequently perform considerably better.

Finally, when the results are considered in the context of the published literature, the proposed system compares favourably. Table 4.4 situates the measured accuracy relative to previously reported Arabic Sign Language studies. It should be noted that these studies employed different datasets and evaluation protocols, so the comparison is indicative rather than strictly controlled.

Table 4.4: Comparison with Prior Arabic Sign Language Studies

Study	Dataset	Approach	Reported Accuracy
Proposed Custom CNN (this thesis)	ArASL, 54,049 images, 32 classes	Custom three-block CNN	94.88%
Noor (2024) - CNN	Custom set, 4,000 images	Hybrid CNN component	94.40%
Noor (2024) - LSTM	Custom set, 4,000 images	Hybrid LSTM component	82.70%
Podder et al. (2023)	Custom RGB video set	MobileNetV2 with LSTM and SelfMLP	87.69%
Das (2023) - ISL	ISL video word dataset	VGG-19 with bidirectional LSTM	87.67%

The proposed system achieves accuracy comparable to the highest figure reported by Noor (2024) while being evaluated on a dataset more than thirteen times larger and covering the complete 32-class Arabic alphabet, which represents a more demanding classification problem than the twenty-word vocabulary used in that study. The result also exceeds the signer-independent benchmark reported by Podder et al. (2023), although it should be acknowledged that signer-independent evaluation constitutes a more challenging protocol than the stratified random partition applied here.

4.8 Real-Time Translation System Evaluation

The best-performing model was integrated into the real-time translation system described in Section 3.12 and evaluated in operational conditions. The system captures webcam frames at 350-millisecond intervals, transmits each frame to the prediction endpoint, and displays the recognized letter together with its confidence score.

In practical operation the system responded at interactive speeds, with the observed end-to-end latency, comprising frame capture, transmission, preprocessing, inference, and display, remaining sufficiently short to sustain a natural signing rhythm. The word-building mechanism successfully assembled sequences of recognized letters into words, and the stabilization logic described in

Section 3.14 proved effective in preventing transient misclassifications during transitions between signs from corrupting the assembled output.

Two practical observations emerged from operational testing. First, recognition reliability is sensitive to lighting conditions and background complexity, since the model was trained on imagery captured under relatively controlled conditions. Second, the confidence score provides useful feedback to the user, since a persistently low confidence value signals that the hand position or framing should be adjusted. These observations inform the limitations and future work discussed in Chapter Five.

Chapter Five: Conclusion and Future Work

5.1 Introduction

This chapter concludes the thesis by summarizing the research undertaken, assessing the extent to which each stated objective was achieved, presenting the principal findings, acknowledging the limitations of the study, and outlining directions for future development.

5.2 Summary of the Research

This research addressed the problem of automated Arabic Sign Language recognition, motivated by the persistent communication barriers faced by Deaf and hard-of-hearing individuals across the Arab world and by the documented shortage of certified sign-language interpreters in many Arab countries.

The study was conducted using the publicly available ArASL dataset comprising 54,049 grayscale images across 32 Arabic alphabet sign classes. A comprehensive comparative benchmark was designed and executed in which seven models were implemented and evaluated under a unified experimental protocol: four traditional machine learning classifiers and three deep learning architectures. All models were trained and validated on the same stratified 80/20 partition using the same random seed, ensuring that observed performance differences are attributable to the architectures themselves.

The proposed Custom CNN, comprising three convolutional blocks with progressive dropout regularization and a dense classification head, achieved the highest validation accuracy of 94.88% while requiring the least training time among the deep learning models. This model was subsequently integrated into a functional real-time translation system built on a FastAPI backend with a browser-based user interface capable of live webcam inference and progressive word assembly.

5.3 Achievement of Research Objectives

Each of the specific objectives stated in Section 1.3 was addressed as follows:

15. Dataset acquisition and preprocessing: Achieved. The ArASL dataset was acquired, verified through a defensive integrity procedure confirming 54,049 valid images with no missing or corrupt entries, and prepared through two distinct preprocessing pipelines appropriate to the deep learning and traditional model families respectively.

16. Implementation of a custom CNN: Achieved. A three-block convolutional architecture was designed, implemented, and trained, attaining 94.88% validation accuracy on standard central processing unit hardware.
17. Implementation of traditional baselines: Achieved. Random Forest, Linear Support Vector Machine, HistGradientBoosting, and Multilayer Perceptron classifiers were implemented and evaluated on the same data partition.
18. Implementation of alternative deep architectures: Achieved. A Light ResNet with residual connections and a MobileNetV2 transfer learning model were implemented and benchmarked against the proposed network.
19. Unified evaluation: Achieved. All seven models were evaluated using validation accuracy and training time under identical conditions, and the learning behaviour of the proposed CNN was analysed through its accuracy and loss curves.
20. Real-time system integration: Achieved. The best-performing model was deployed within a functional client-server application providing live webcam recognition and word assembly through a browser-based interface.

5.4 Key Findings

- Deep learning substantially outperforms traditional machine learning for Arabic Sign Language image classification, with the proposed CNN exceeding the strongest traditional classifier by 14.63 percentage points.
- A compact, purpose-designed convolutional network can outperform a substantially larger pretrained model. The proposed CNN exceeded MobileNetV2 transfer learning in accuracy while requiring approximately 37% of its training time, a consequence of the domain mismatch between ImageNet natural photography and low-resolution grayscale sign imagery.
- Additional architectural depth does not guarantee improved performance. The Light ResNet achieved only 80.97%, below both other deep architectures, indicating that residual complexity provided no measurable benefit on this dataset within the applied training budget.
- Algorithm selection matters considerably within the traditional machine learning family, with performance ranging from 47.85% for Linear SVM to 80.25% for Random Forest on identical data.
- The close alignment between training and validation curves confirms that the combination of dropout regularization and data augmentation effectively mitigated overfitting despite the relatively large capacity of the dense classification head.

- The trained model is sufficiently lightweight to support real-time inference at 350-millisecond intervals within a browser-based application, confirming the practical feasibility of the proposed approach.

5.5 Limitations of the Study

- **Scope restricted to the static alphabet:** The system recognizes the 32 individual letters of the Arabic manual alphabet. It does not address dynamic word-level signs, continuous signing, or full sentence-level translation, all of which require temporal modelling across sequences of frames.
- **Single-source dataset:** All imagery originates from the ArASL corpus, which was collected in a single geographic region under relatively controlled conditions. Generalization to different signers, lighting environments, backgrounds, and regional variants of Arabic Sign Language has not been empirically verified.
- **Evaluation protocol:** A stratified random partition was used, which permits images from the same signer to appear in both the training and validation sets. A signer-independent protocol, in which no signer appears in both partitions, would provide a more stringent test of generalization.
- **Constrained training budget:** All deep learning models were trained for five epochs on central processing unit hardware. Extended training with graphics processing unit acceleration would likely improve results further, and may particularly benefit the deeper residual architecture.
- **Absence of speech output:** The system produces recognized letters and assembled words as on-screen text. An audible speech synthesis stage, which would allow a hearing interlocutor to receive the message aurally, has not been implemented.
- **Limited metric coverage:** Evaluation focused on validation accuracy and training time. Per-class precision, recall, and F1-score, together with confusion matrix analysis, would provide finer-grained insight into which specific Arabic signs are most frequently confused.

5.6 Future Work

21. **Integration of Arabic speech synthesis:** Adding a text-to-speech stage would complete the sign-to-speech communication loop, allowing recognized words to be delivered audibly to a hearing interlocutor. This represents the most direct extension of the present system.
22. **Extension to dynamic and continuous signing:** Incorporating temporal architectures such as convolutional networks combined with long short-term memory layers, or transformer-

based sequence models, would enable recognition of dynamic word-level signs and continuous signing rather than isolated letters.

23. Signer-independent evaluation: Re-partitioning the dataset by signer identity and repeating the benchmark would provide a more rigorous assessment of generalization and would permit direct comparison with signer-independent studies such as Podder et al. (2023).
24. Expansion of dataset diversity: Collecting additional imagery covering varied lighting conditions, backgrounds, camera angles, and regional dialectal variants of Arabic Sign Language would improve robustness in uncontrolled deployment environments.
25. Extended training with hardware acceleration: Repeating the benchmark with graphics processing unit acceleration and a longer training schedule would establish whether the observed ranking of architectures persists under a larger computational budget.
26. Comprehensive per-class analysis: Computing per-class precision, recall, and F1-score together with a confusion matrix would identify which Arabic signs are most frequently confused, enabling targeted data augmentation or architectural refinement for those specific classes.
27. Mobile deployment: Porting the trained model to a mobile application through frameworks such as TensorFlow Lite would substantially improve accessibility, allowing the system to be used on a smartphone without dependence on a server connection.

Appendices

Appendix A: Dataset Information

Property	Value
Dataset name	Arabic Alphabets Sign Language (ArASL)
Reference	Latif, Mohammad, Alghazo, AlKhalaf and AlKhalaf (2019)
Total images	54,049
Classes	32 Arabic alphabet signs
Resolution	64 x 64 pixels, grayscale
Metadata file	ArSL_Data_Labels.csv
Training samples	43,239 (80%)
Validation samples	10,810 (20%)
Integrity check result	54,049 valid; 0 missing; 0 corrupt

Appendix B: Model Validation Accuracy Results

Model	Type	Validation Accuracy	Training Time (s)
Custom CNN	Deep Learning	94.88%	103.54
MobileNetV2 (TL)	Deep Learning	93.64%	280.73
Light ResNet	Deep Learning	80.97%	152.44
Random Forest	Traditional ML	80.25%	1.90
HistGradientBoosting	Traditional ML	77.75%	237.27
Multilayer Perceptron	Traditional ML	61.60%	18.72
Linear SVM	Traditional ML	47.85%	210.60

Appendix C: Software and Development Environment

Component	Specification
Programming language	Python 3
Deep learning framework	TensorFlow / Keras
Machine learning library	scikit-learn
Image processing	OpenCV, Pillow
Numerical computing	NumPy
Visualization	Matplotlib
Web framework	FastAPI with Uvicorn
Front end	HTML5, CSS3, JavaScript

Component	Specification
Training hardware	CPU-based environment

Appendix D: CNN Architecture Summary

Layer	Configuration	Output
Input	Grayscale image	64 x 64 x 1
Convolution 1	16 filters, 3 x 3 kernel, ReLU	62 x 62 x 16
Max Pooling 1	2 x 2 window	31 x 31 x 16
Dropout 1	Rate 0.25	31 x 31 x 16
Convolution 2	32 filters, 3 x 3 kernel, ReLU	29 x 29 x 32
Max Pooling 2	2 x 2 window	14 x 14 x 32
Dropout 2	Rate 0.25	14 x 14 x 32
Convolution 3	64 filters, 3 x 3 kernel, ReLU	12 x 12 x 64
Max Pooling 3	2 x 2 window	6 x 6 x 64
Dropout 3	Rate 0.25	6 x 6 x 64
Flatten	-	2,304
Dense 1	128 units, ReLU	128
Dropout 4	Rate 0.5	128
Output	32 units, Softmax	32

Appendix E: Project File Structure

File	Description
train_and_compare.py	Unified benchmarking script training and evaluating all seven models
train_and_save_model.py	Training and serialization of the final production model
evaluate_all_models.py	Evaluation utilities and metric computation
app.py	FastAPI backend exposing the prediction endpoint
run_app.py	Application launcher with dependency verification
index.html	Browser-based user interface
arasl_model.keras	Serialized trained CNN model
class_names.json	Ordered mapping of class indices to sign labels
benchmark_report.md	Automatically generated benchmarking results
requirements.txt	Python dependency specification

Appendix F: Required Dependencies

The following Python packages are required to reproduce the experiments and to run the deployed application:

- tensorflow - deep learning framework used to define, train, and serialize the models
- scikit-learn - traditional machine learning classifiers, data partitioning, and label encoding
- numpy - numerical array operations and in-memory dataset handling
- pandas - metadata parsing and tabular data manipulation
- opencv-python - image decoding, colour conversion, and resizing
- pillow - image integrity verification during dataset preparation
- matplotlib - generation of accuracy, loss, and comparison plots
- fastapi and uvicorn - web service framework and application server
- python-multipart - handling of uploaded image frames

References

[Bani Baker, 2023]

Bani Baker, R. (2023). Image-based Arabic sign language recognition system using transfer deep learning models. *Applied Computational Intelligence and Soft Computing*, 2023, 5195007. <https://doi.org/10.1155/2023/5195007>

[Chollet, 2015]

Chollet, F. (2015). Keras: The Python deep learning API. Retrieved from <https://keras.io>

[Das, 2023]

Das, S. (2023). A deep sign language recognition system for Indian sign language. *Neural Computing and Applications*, 1469-1481. <https://doi.org/10.1007/s00521-022-07840-y>

[He et al., 2016]

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770-778.

[Hussain, 2023]

Hussain, A. (2023). Advancements in Indian sign language recognition systems: Enhancing communication and accessibility for the deaf and hearing impaired. *Asian Journal of Engineering Sciences*, 12(2).

[Kingma & Ba, 2015]

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the International Conference on Learning Representations*.

[Kumar, 2023]

Kumar, R. (2023). MediaPipe and CNNs for real-time ASL gesture recognition. *arXiv preprint arXiv:2305.05296*.

[Latif et al., 2019]

Latif, G., Mohammad, N., Alghazo, J., AlKhalaf, R., & AlKhalaf, R. (2019). ArASL: Arabic alphabets sign language dataset. *Data in Brief*, 23, 103777. <https://doi.org/10.1016/j.dib.2019.103777>

[Noor, 2024]

Noor, T. H. (2024). Real-time Arabic sign language recognition using a hybrid deep learning model. *Sensors*, 24(11), 3683. <https://www.mdpi.com/1424-8220/24/11/3683>

[Pedregosa et al., 2011]

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.

[Podder et al., 2023]

Podder, K. K., Ezeddin, M., Chowdhury, M. E. H., Sumon, M. S. I., Tahir, A. M., & Ayari, M. A. (2023). Signer-independent Arabic sign language recognition system using deep learning model. *Sensors*, 23(16), 7156. <https://doi.org/10.3390/s23167156>

[Sandler et al., 2018]

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

[Sharma, 2025]

Sharma, G. (2025). Real-time sign language recognition and translation using MediaPipe and random forests for inclusive communication. *IEEE Xplore*.

[Srivastava et al., 2014]

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929-1958.