



**College of Informatics
Midocean University
UAE**

Research Title

**Deep Learning Techniques for Arabic
Sign Language Recognition**

Prepared By

Sultan Ali Senan

University ID number: 253002563

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Artificial Intelligence

Supervisor: Dr. Amjad Qtaish

College of Informatics, Midocean University

Academic year

2026

Table of Contents

Table of Contents	ii
Acknowledgements	iv
Abstract	v
List of Abbreviations	vii
List of Figures	viii
List of Tables	ix
Chapter One: Introduction	1
1.1 Overview	1
1.2 Problem Definition.....	2
1.3 Thesis Objectives	2
Main Objective.....	2
Specific Objectives	3
1.4 Thesis Questions	3
1.5 Thesis Contribution.....	3
1.6 Thesis Organization	4
Chapter Two: Related Work	6
2.1 Arabic Sign Language Recognition	6
2.2 Deep Learning Techniques for Sign Language Recognition	7
2.3 Speech Recognition and Speech Synthesis Systems	8
2.4 Interactive Sign Language Translation Systems	9
2.5 Machine Learning and Computer Vision Approaches for Sign Language Recognition	9
2.6 Comparative Studies and Benchmarking Practice	10
2.7 Research Gap	12
Chapter Three: Methodology	14
3.1 Introduction.....	14
3.2 Dataset Collection and Preparation.....	14
3.3 Data Preprocessing.....	15
Preprocessing for Deep Learning Models.....	15
Preprocessing for Traditional Machine Learning Models	15
3.4 Regularization Strategy.....	16
3.5 Proposed CNN Architecture	17
3.6 Traditional Machine Learning Models	18
3.7 Deep Learning Models.....	19

3.8 Model Training Procedure	20
3.9 Performance Evaluation Metrics.....	21
3.10 Training and Validation Analysis	22
3.11 Comparative Benchmarking Procedure	23
3.12 Interactive Translation System Architecture	23
3.13 User Interface Design	24
3.14 System Workflow	25
Chapter Four: Results and Discussion	27
4.1 Introduction.....	27
4.2 Experimental Setup.....	27
4.3 Comparative Model Performance Analysis	27
4.4 Accuracy Comparison Results.....	28
4.5 Per-Class Performance Analysis.....	30
4.6 Confusion Matrix Analysis.....	31
4.7 Discussion of Results.....	32
4.8 Deployed System Evaluation.....	34
Chapter Five: Conclusion and Future Work	36
5.1 Introduction.....	36
5.2 Summary of the Research	36
5.3 Achievement of Research Objectives	36
5.4 Key Findings.....	37
5.5 Limitations of the Study.....	38
5.6 Future Work	39
Appendices.....	41
Appendix A: Dataset Information.....	41
Appendix B: Model Validation Accuracy Results.....	41
Appendix C: Software and Development Environment	41
Appendix D: CNN Architecture Summary.....	42
Appendix E: Project File Structure	42
Appendix F: Required Dependencies	42
References.....	44

Acknowledgements

It is my pleasure to express my sincere thanks to all those who have helped me complete this research project successfully.

First of all, I would like to thank my supervisor, Dr. Amjad Qtaish, for providing me with necessary advice, support, and constructive criticism during the process of conducting this study.

Next, I would like to express my gratitude to the members of faculty and academic staff of the College of Informatics of Midocean University for providing me with necessary information, resources, and academic environment to conduct this research project.

In addition, I would like to thank my family and friends for their understanding, support, and encouragement during this study.

Last but not least, I should thank the authors of the publicly available Arabic Alphabets Sign Language (ArASL) database, without whose contribution the experimental part of this research could not be conducted.

Abstract

Barriers to effective communication between the Deaf and the hearing population represent one of the persisting issues of social and technological nature in the Arab world. Despite the fact that sign languages represent the predominant form of communication within Deaf communities, the task of automatic translation between sign languages and written or spoken languages is still quite challenging computationally-wise. In addition, Arabic Sign Language (ArSL) is spoken throughout more than twenty Arab countries, however, has attracted significantly less attention of artificial intelligence researchers as compared to ASL or BSL, and a severe deficit of qualified sign-language interpreters prevails in many Arab countries.

This study discusses the development, implementation, and evaluation of a deep learning-based system for recognition of Arabic Sign Language. It is based on the open-source dataset of Arabic Alphabets Sign Language (ArASL), containing 54,049 gray-scale images belonging to 32 Arabic alphabet sign classes. Comparative benchmarking study has been carried out, implementing and evaluating seven different machine learning and deep learning models under the same experimental protocol: four 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 blocks, and MobileNetV2 transfer learning).

The results of experiments show significant superiority of deep learning architectures over traditional machine learning methods when dealing with image-based sign recognition. The proposed custom CNN showed the best validation accuracy of 94.88%, followed by MobileNetV2 transfer learning model with 93.64% and Light ResNet with 80.97%. Traditional classifiers showed better performance of Random Forest with 80.25% and poorer of Linear SVM with 47.85%. At that, it should be noted that the proposed custom CNN not only showed higher accuracy, but also trained significantly faster, requiring 37% of time needed for the transfer learning model.

The best-performing model has been implemented in an interactive translation system based on the FastAPI backend and browser interface. The developed system captures camera frame in 350 milliseconds, recognizes the observed hand sign, and gradually collects the letters into words that

are shown to the user. The result of this research is a working, accessible, and reproducible communication pipeline for narrowing the gap between the Deaf and the hearing population in the Arab world, as well as the comparison study documenting a reference baseline for future research in Arabic assistive artificial intelligence.

Keywords: *Arabic Sign Language; Deep Learning; Convolutional Neural Networks; Image Classification; Model Benchmarking; Interactive 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	15
3.3-1	Data Preprocessing	16
3.5-1	Proposed CNN Architecture	18
3.5-2	Implementation of the Proposed CNN Architecture	18
3.6-1	Traditional Machine Learning Models	19
3.7-1	Deep Learning Models	20
3.8-1	Model Training Procedure	21
3.9-1	Performance Evaluation Metrics	22
3.10-1	Training and Validation Analysis	22
3.11-1	Comparative Benchmarking Procedure	23
3.12-1	Interactive Translation System Architecture	24
3.12-2	Implementation of the Interactive Translation System	24
3.13-1	User Interface Design	25
4.3-1	Model Validation Accuracy Comparison	29
4.6-1	Confusion Matrix of the Proposed CNN	32

List of Tables

Table	Title	Page
2.1	Comparative Summary of Related Work in Sign Language Recognition	11
3.1	Summary of the ArASL Dataset	14
3.2	Configuration of Traditional Machine Learning Models	18
3.3	Configuration of Deep Learning Architectures	20
3.4	Training Hyperparameters	20
4.1	Dataset Split Summary	27
4.2	Validation Accuracy and Training Time of All Evaluated Models	28
4.3	Accuracy Relative to Training Cost for Deep Learning Models	30
4.4	Per-Class Precision, Recall and F1-Score on the Validation Partition	30
4.5	Comparison with Prior Arabic Sign Language Studies	34

Chapter One: Introduction

1.1 Overview

Despite the fact that communication is one of the basic needs of humans, millions of deaf and hard of hearing persons all over the world face difficulties in their daily communication due to accessibility problems. Though the automatic translation from sign language into spoken language is one of the biggest technical challenges, the sign language still is one of the main ways of communication in the deaf communities. In the Arab world, Arabic Sign Language (ArSL) is used in more than twenty countries but it is significantly understudied in the scope of artificial intelligence in comparison with such widely studied sign languages as ASL and BSL.

The recent breakthrough in deep learning changed the fields of computer vision, speech processing, and natural language understanding. Convolutional neural networks, recurrent neural networks, LSTM networks, and transfer learning models showed excellent results in recognizing complicated visual patterns. It makes deep learning methods particularly suitable for sign language recognition task which requires precise understanding of hand shape, finger configuration and spatial orientation.

Deep learning-based sign language recognition system for Arabic Sign Language is an attempt to solve the problem of communication by converting the gestures made by hands into readable text understandable by those who do not know the sign language. The proposed system includes the processes of image acquisition, preprocessing, feature extraction, classification and output generation. With the help of deep learning algorithms, the proposed system automatically learns discriminative visual features of signs from images with minimum engineering effort.

The development of such system is not only an important technological but also social contribution. It provides support for assistive communication technologies, increases accessibility and inclusiveness, and helps to create Arabic sign language digital resources. Additionally, the creation of documented and repeatable benchmarking pipeline for ArSL recognition paves the way for future researches in Arabic assistive artificial intelligence which is rather understudied.

To achieve high level of accuracy, responsive performance and usability for practical communication purposes, this project is aimed at design and implementation of intelligent sign language recognition system based on deep learning.

1.2 Problem Definition

Though there have been various advancements in the field of technologies recognizing sign languages, most of the existing systems mainly focus on popular sign languages like American or British Sign Languages. The technological development supporting Arabic Sign Language is relatively less than that of other sign languages and hence the recognition systems along with the evaluation procedures and benchmarking studies are also rare.

The deaf population in the Arab countries has major problems in communicating due to the non-availability of any automated recognition systems for Arabic Sign Language and hence it becomes difficult for them to avail the available public services, employment and educational opportunities. Moreover, the scarcity of qualified sign-language interpreters in several Arab countries makes the problem even harder as human interpretation is not always available whenever it is necessary.

This is yet another research gap that needs to be resolved. There is a scarcity of the studies that conduct a comparative analysis of the performance of the traditional machine learning approaches versus deep learning architectures for the classification of Arabic Sign Language data under the same set of experimental conditions. Such studies will help in determining whether deep learning is superior than machine learning in terms of classification accuracy, time and computational complexity.

Hence, there is a need of developing an intelligent, accurate and computationally efficient system that uses deep learning techniques to recognize Arabic sign gestures and translate them into the text. This requires a comparative study of different model architectures which can perform this task effectively.

1.3 Thesis Objectives

Main Objective

For designing, implementing and evaluating a recognition system for Arabic Sign Language using deep learning techniques.

Specific Objectives

1. For the acquisition, organization and pre-processing of a large set of Arabic Sign Language alphabet gestures appropriate for supervised deep learning.
2. For the implementation and training of a novel CNN architecture for Arabic sign recognition using standard computational equipment.
3. For the implementation of conventional machine learning baselines such as Random Forest, Linear Support Vector Machine, HistGradientBoosting and Multilayer Perceptron.
4. For the implementation of additional deep learning architectures including a Light ResNet with residual connections and MobileNetV2 transfer learning for comparison of the proposed CNN against other deep architectures.
5. For the assessment of all the models based on the same protocol of evaluation of validation accuracy and training time, as well as for analysis of the learning behaviour of the proposed CNN through its recorded accuracy and loss values.
6. For incorporating the best performing model into an interactive translation system that uses the browser interface to form words using identified letters.

1.4 Thesis Questions

- How can the accuracy of Arabic Sign Language Recognition be enhanced through deep learning methods?
- How do deep learning models compare with machine learning models with respect to accuracy when tested on the same Arabic Sign Language data set under similar experimental settings?
- Which model has the best tradeoff among accuracy, training time, and feasibility?
- Would a customized CNN perform better than transfer learning on grayscale sign language images?
- Can such a system work at interactive speeds to make it usable in real-life communication situations?

1.5 Thesis Contribution

The main contributions of this thesis in the fields of assistive technology and Arabic Sign Language recognition include:

- **Comprehensive comparative benchmarking:** This thesis applies seven different machine learning and deep learning algorithms to the same Arabic Sign Language dataset using one common set of experiments to create a benchmarking reference that is quite rare in the literature about ArSL.
- **A validated custom CNN architecture:** The designed lightweight CNN model reaches 94.88% validation accuracy, which surpasses not only the accuracy of conventional classification models but also that of deep learning architectures such as MobileNetV2 transfer learning, while needing much less time for training.
- **Illustration of trade-offs between efficiency and accuracy:** This thesis demonstrates that a compact architecture tailored for specific needs performs better than pretrained heavy architecture in grayscale images of the domain of interest.
- **Interactive translation prototype:** Unlike the hypothetical prototypes in theoretical approaches, the approach in this research yields a fully functional and real-time end-to-end system capable of performing webcam inference and assembling recognized signs into words using an online web application.
- **Accessibility and social implications:** The developed prototype is an actual communication system that is aimed at assisting people who are deaf or hearing impaired in communicating in the Arab world.
- **Reproducibility:** Source code, model artefacts, dataset preparation pipeline, and auto-generated benchmarking reports are available for review, enabling reproduction and extension of the reported results.

1.6 Thesis Organization

Problem, methodology, implementation, and outcomes of the proposed deep-learning based Arabic sign language recognition system are discussed in the next few chapters:

Chapter one introduces the background of the study, the communication problems faced by the Deaf community, and importance of an automated Arabic Sign Language Recognition System.

The second chapter gives a thorough review of the literature on recognition of sign languages, architectures of deep learning networks for classifying images, speech and language processing systems, real-time translation systems, and computer vision techniques. The chapter ends with the identification of the particular research gap that the thesis is trying to fill.

The third chapter describes the methodology of the research, which includes data collection and preparation, preprocessing techniques, the architecture of the proposed convolutional neural network, settings of all benchmark models, training process, performance metrics, and the design of the interactive translation system with its user interface.

In the fourth chapter, the results of the experiments are presented and analyzed in terms of the comparative performance of all seven investigated models, behavior of the proposed CNN during training and validation, interpretation of the results, and assessment of the implemented interactive system.

The fifth chapter is devoted to summarizing the research, evaluating how each research objective was met, discussing the main findings, pointing out the limitations of the research, and offering directions for future work.

Chapter Two: Related Work

2.1 Arabic Sign Language Recognition

Of course, communication is a complicated task for people who are Deaf or hard of hearing. They communicate using sign language between themselves and the rest of the population. However, there are many members of the population who can communicate only orally and in writing and do not know sign language. It means that the creation of technical tools for automatic sign language interpretation is highly necessary.

Arabic Sign Language (ArSL) is a gestural visual language used by members of the Deaf communities in the Arab world. Similar to other sign languages, ArSL conveys meaning using hand shape, hand orientation, movement and facial expression and the grammar of this language is very different from spoken Arabic. Due to the differences between variations of Arabic sign language in different Arab countries, developing standard recognition resources is a challenging task and has been historically slower than developing standardized recognition resources for more centralized sign languages.

An important contribution to the field was provided by Latif, Mohammad, Alghazo, AlKhalaf, and AlKhalaf (2019). This paper introduces a new Arabic Alphabets Sign Language (ArASL) dataset. It includes 54,049 gray scale images of 32 different Arabic alphabet signs recorded from more than forty participants in Saudi Arabia and resized to 64 x 64 pixels. Before the creation of this dataset, researchers working on the Arabic Sign Language recognition used private and limited datasets, and therefore it was impossible to reproduce the results or compare them with each other. ArASL dataset became a common benchmark for the future researches in this area and was chosen for the experimental part of this thesis.

Developing this idea further, Noor (2024) proposes a hybrid deep learning model for real time recognition of Arabic sign language. In his model, the author uses a convolutional network to extract spatial features and long short-term memory network to process temporal sequence. Tested on a custom dataset with 4,000 images of ten dynamic and ten static gesture words, convolutional network demonstrates 94.40% accuracy and long short-term memory network shows 82.70%. The motivation for the paper was defined as the lack of certified sign language interpreters in Saudi Arabia, which is the same concern of the current research.

Podder et al. (2023) worked with a more complex variant of the task, which is signer-independent recognition of gestures. It requires the system to be able to recognize gestures from people whose gestures were not seen during the training process. The authors use MobileNetV2 and ResNet18 architectures as a convolutional backbone, combine them with a long short-term memory layer and a custom self-attention multilayer perceptron module and evaluate them on RGB video with face and hand segmentation. With the best configuration, their system demonstrates 87.69% accuracy, 88.57% precision, 87.69% recall and 87.72% F1-score, which is eleven percentage points higher than any previous results on this type of dataset.

Finally, Bani Baker (2023) studies the performance of transfer learning for image-based Arabic alphabetic sign recognition. In his work, he compares several pretrained convolutional architectures fine-tuned on Arabic sign imagery with equivalent networks trained from scratch. As a result, it was shown that transfer learning usually outperforms training from scratch if the dataset size is limited. It provides a good background for the future experiment, as in the current paper I test whether the same is true for a large dataset of more than fifty thousand grayscale images.

2.2 Deep Learning Techniques for Sign Language Recognition

Current sign language recognition systems are constructed almost exclusively using deep learning techniques. Convolutional Neural Networks are especially suitable for this task as they learn hierarchical representations from pixels. Earlier convolutional layers react to primitive features like edges and gradients, while deeper layers progressively assemble higher level representations corresponding to different finger configurations and shapes. Thus, the hierarchy takes away the need for hand-crafted feature engineering that was characteristic for earlier methods of gesture recognition.

The Residual networks, which were proposed to solve the problem of degradation during training of very deep models, use skip connections, through which gradients can propagate directly. Although residual architectures turned out to be extremely useful for solving the problem of image classification on large and high-resolution datasets, this technique does not show much gain on smaller and lower-resolution domain-specific datasets. Therefore, the current study proposes a Light ResNet as an additional architecture for testing this hypothesis on Arabic signs.

Finally, Transfer Learning is a third strategy that uses pre-trained on a generic large dataset network like ImageNet and trains its classifier for a new task. MobileNetV2 uses depthwise and separable convolutions and inverted residual blocks to achieve efficiency of inference on mobile devices. However, the value of the transfer learning approach depends on the similarity between the two datasets. Since grayscale images used in the problem differ drastically from natural color images that were used to train the ImageNet, it is worth examining the transferable benefits of the technique empirically in Chapter Four.

Recurrent Architectures like Long Short-Term Memory networks are usually used together with convolutional layers for modelling temporal dependencies when working with dynamic and continuous signing. As the problem solved in this study is related to static signs where one image is sufficient to capture all the necessary information about the sign, only convolutional architectures are required. Temporal dependency modelling is identified as a way to extend the system in Chapter Five.

2.3 Speech Recognition and Speech Synthesis Systems

Automatic Speech Recognition is another important field of study relevant for multilingual and voice-driven human-computer interaction. The speech recognition systems transform the audio input into text while speech synthesis systems transform the text input into spoken audio output. Together these technologies form the audio analogue of the visual pipeline used in this study.

Both problems have specific difficulties associated with the Arabic language. It has a significant dialectal diversity, a complex morphological system, and it is traditionally not written with short vowels marked. These properties historically made Arabic speech technology harder to develop in comparison to the analogues for languages with simpler orthography and morphology.

Concerning the development of assistive technologies for Deaf people, the usage of speech synthesis was often suggested as the final stage for producing recognized signs into sounds understandable by a hearing interlocutor who does not know sign language. There are several examples of sign language recognition systems including such a stage in their workflow. The system presented in this thesis generates recognized letters and assembled words in the form of on-screen text instead of speech; thus, the implementation of Arabic speech synthesis as a future step is clearly outlined in Section 5.6.

2.4 Interactive Sign Language Translation Systems

It goes without saying that a recognition model can be of any use only if it can be integrated into a system able to operate interactively. In case of real-time sign language translation systems, the video frames should be acquired by the camera, processed, inferenced, and delivered to the user in time necessary for sustaining natural communication.

There are several design paradigms existing in this area. Client-server approaches put the trained model behind the web service interface with a browser-based front end to capture the frames and send them to the inference. Such a solution has advantages of model centralization, easier updates, and no necessity to install special software on the end-user side. On the other hand, the inference may be performed either fully on the client-side or on embedded hardware; this way has disadvantages of network dependency but allows making the system more flexible.

Kumar (2023) demonstrated an architecture of real-time recognition of American Sign Language which involved MediaPipe hand landmarks extraction followed by a convolutional classifier; the model achieved the letter recognition accuracy up to 99.95%. Sharma (2025) showed that combining landmarks with the random forest classifier allows reaching high accuracy at significantly lower computational costs; such an approach suits perfectly devices with low processing power. Das (2023) solved the task of recognizing word-level Indian Sign Language by using the combination of VGG-19 backbone and a bidirectional LSTM network; the average accuracy of such a model is 87.67%.

A common aspect in continuous alphabet-based systems is the combination of letter predictions into whole words. Since the signer keeps each letter for some time, subsequent frames result in predictions from the same class. Therefore, practical systems have some form of temporal stabilization logic in order to commit to a letter only after the prediction has been stable for a certain period of time. This is the case for the system designed in this thesis work, as explained in Section 3.14.

2.5 Machine Learning and Computer Vision Approaches for Sign Language Recognition

The classic approach to sign language recognition before deep learning was the combination of classical computer vision and traditional machine learning classifiers. The general pipeline would

consist of a skin colour segmentation step to separate the region containing the hand, a feature extraction stage based on hand-engineered feature representations such as histograms of oriented gradients or scale-invariant features, and a classification phase utilizing algorithms such as support vector machines or decision tree ensembles.

While being outdated, the classical methods of recognition are important for the baseline comparison purposes. Showing how well a properly configured classical classifier can do allows assessing the added value brought by deep learning. Random Forests represent a special case of decision trees which average multiple classifiers' output from bootstrapped training sets. This results in especially robust models which don't require much hyperparameter optimization. Similarly, gradient boosting algorithms build multiple decision trees and train them in sequence so that each subsequent classifier can fix mistakes made by the previous one.

Support vector machines aim to find a hyperplane with the maximal distance between two classes. However, their performance heavily relies on the linear separability of feature space. For example, linear SVM applied to the flat pixel space of image data is likely to perform very badly due to lack of separability of that feature space. On the other hand, multilayer perceptron falls somewhere between SVMs and CNNs by its ability to learn complex decision boundaries while missing spatial inductive bias inherent to convolutional architectures.

Dimensionality is the major limitation of the classical classifiers. Flattening a 64x64-pixel image results in 4,096 dimensional feature space and running such classifiers on tens of thousands of samples is computationally and memory-wise expensive. Common ways of dealing with this problem include stratified training samples subsampling and spatial downsampling of images. This technique is employed in the experimental procedure described in Chapter Three.

2.6 Comparative Studies and Benchmarking Practice

Comparison studies are important in maturing any research field. Publications often report results of a particular suggested architecture using a privately gathered dataset in different experimental settings. This makes it difficult to compare results in different publications because the accuracy results can vary due to differences in the data used, preprocessing or even the experimental setup.

According to Hussain (2023), in studying the Indian Sign Language recognition literature, the distribution of the attention and evaluation practice is skewed to some regional sign languages,

and the lack of comparisons impedes further developments. This same pattern applies to Arabic Sign Language.

For a comparison to be done correctly, all the models being compared should be trained and tested on the same dataset using the same partition and metrics on the same machine. In this case, differences in the result will be due to the architecture itself. It is also important to consider the training time in addition to the accuracy, where the model whose accuracy is slightly higher but requires several times more training effort may be less favorable for implementation. This experiment design is implemented in this thesis where seven architectures are compared using a unified protocol described in Chapter Three.

Table 2.1 below presents the relevant literature comparison.

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-	ASL gesture image dataset	MediaPipe landmark	Up to 99.95% letter	Validated on ASL rather than ArSL

Reference	Title / Description	Dataset	Methods	Reported Results	Limitations
	Time ASL Gesture Recognition		extraction with CNN classifier	recognition accuracy	
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

Though the above developments have been made in the fields discussed in the previous sections, most of the existing literature is concerned with the popularly researched sign languages like ASL and ISL. Research conducted in the field of ArSL has been relatively less. There has not been much effort to automate recognition and translation in Arabic Sign Language.

In particular, there are three reasons why the current work is important. Firstly, even though a few individual models have been suggested for the problem of ALS recognition, direct comparison of classical machine learning models against a variety of deep learning models under the same experimental conditions on the same standardized dataset is not common practice. This makes it impossible to understand the actual difference between the two approaches in terms of their efficiency in solving the task.

Secondly, while the accuracy of various models is almost always stated, computational efficiency in terms of the training time is not usually provided. However, when developing an assistive technology for daily use, training time, inference latency and hardware constraints become crucial factors to take into account.

Thirdly, a lot of work presented in scientific literature ends at the point where models are evaluated and never leads to a practical solution. Indeed, the gap between a model presented in tabulated form and the application that deaf people can use is quite significant and needs further development.

This thesis fills all three gaps mentioned above. The work provides a controlled benchmark of seven different models trained on the ArASL dataset consisting of 54,049 images. The accuracy and training time for each model are stated. Moreover, the architecture providing the best results is used to implement a real-life interactive translation system.

Chapter Three: Methodology

3.1 Introduction

This chapter discusses the methodology implemented for the design, creation, and testing of the proposed recognition system for Arabic Sign Language. It includes the description of the dataset used, the pre-processing methods used, the regularization techniques implemented, the architecture of the convolutional network proposed, the configuration of each baseline model that is used as part of the comparative benchmark, the training procedure, the evaluation criteria, and the architecture of the interactive system deployed.

This methodology relies on one key premise: the models should be trained and tested under exactly the same conditions, in order to attribute any differences in performance to the architecture itself rather than to the processing and evaluation procedure. This has led to the creation of a common pipeline in which the dataset loading and partitioning process and the evaluation metrics calculation procedure are identical for all seven models.

3.2 Dataset Collection and Preparation

The experiments are performed using the Arabic Alphabets Sign Language (ArASL) dataset published by Latif et al. (2019). The dataset consists of 54,049 gray-scale images depicting 32 letters of the Arabic manual alphabet, gathered from more than 40 signers. All the images are of the same resolution (64*64 pixels), and are provided in a single-channel gray-scale format which eliminates the necessity of re-sizing in the pre-processing step.

The image is located in a directory with the name of its class and there is also a metadata file – ArSL_Data_Labels.csv – containing filename and class label for each image. The 32 classes include the letters of the Arabic alphabet as well as al and la, two composite 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

Property	Value
Number of classes	32
Image resolution	64 x 64 pixels
Colour format	Grayscale (single channel)
Number of signers	More than 40 participants
Metadata file	ArSL_Data_Labels.csv

A data integrity procedure to detect any anomalies was carried out before the experiments. The recursive directory scan found the absolute path of all images based on their filenames. After that, each metadata entry was checked against the path map and verified with three conditions: existence of the file on disk, nonzero size, and ability to open and decode the image as a valid file. There were no missing files or corrupt entries found; 54,049 clean image paths have been verified. Integer class labels were encoded to integers from 0 to 31.



Figure 3.2-1: Dataset Collection and Preparation

3.3 Data Preprocessing

Preprocessing needs vary greatly for the deep learning architectures and traditional classifiers, thus two different preprocessing pipelines have been implemented.

Preprocessing for Deep Learning Models

The entire set of 54,049 images is preloaded in memory as a single array of unsigned eight-bit integers that take up roughly 211 megabytes of memory. This way, there will be no repeated disk accesses and decoding of images in every training epoch. A TensorFlow data pipeline was created from the in-memory array and converts the images to floating point numbers and scales the pixel intensities to range [0,1] on per-batch level. The data pipeline applies shuffling with buffer of 10,000 samples, mini-batching with batch size of 64, and prefetches the data to overlap with computations.

Preprocessing for Traditional Machine Learning Models

Traditional classifiers need to have the images reshaped to the one dimensional feature vector which adds some constraints to data preprocessing. The dataset, in whole, would create matrix of $54,049 \times 4,096$ features that make up roughly 221 million features which is too many for the algorithms like support vector machines which are unable to handle such number of features in memory. Thus, two usual mitigation methods have been applied to the problem. First, random subset of 5,000 training samples and 2,000 validation samples has been taken from the dataset. Since the subset is large enough to cover 32 classes, it preserves the distribution of the classes approximately (although it is not stratified). Second, the images have been downsampled spatially to 32 by 32 pixels from 64 by 64 using array step slicing. This operation reduced feature space from 4,096 to 1,024 for every sample. The pixel values are scaled to range $[0,1]$.

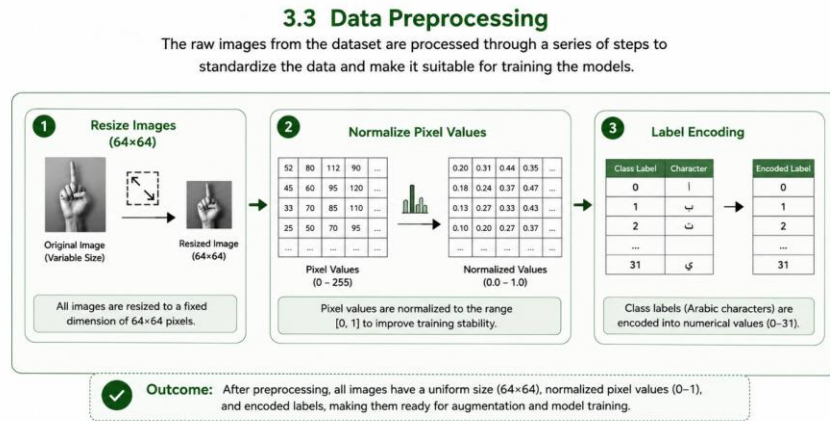


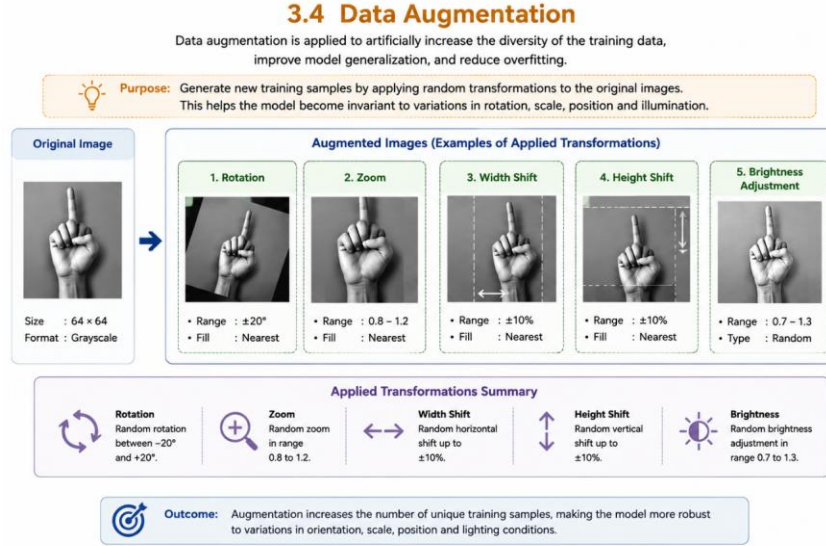
Figure 3.3-1: Data Preprocessing

3.4 Regularization Strategy

The regularization of the convolutional models was done through dropout. Data augmentation was not applied to the data; the images of the training partition were taken precisely as provided in the ArASL corpus, without applying rotation, translation, scaling, or photometric transformations. This is intentional, considering the design requirement stated in Section 3.5 to train the model quickly on central processing unit hardware since data augmentation layers increase computations per batch without decreasing the number of epochs.

There are two properties of the ArASL corpus reducing the necessity of synthetic augmentation of data in the current context. The first property is the size of the training partition, 43,239 images across 32 classes, which provides a substantial number of examples per class for a compact network of 322,464 parameters. The second property is the presence of certain variability in the

images in terms of lighting differences both between and within classes, as stated in Section 3.2. The lack of augmentation, however, still can be considered the weakness of the approach, because it means that the model is not trained specifically to recognize variations in rotation, scale or light. This fact explains the sensitivity of the model to lighting changes and background during testing operationally, discussed in Section 4.8. Data augmentation is one of the directions of future work mentioned in Section 5.6.



3.5 Proposed CNN Architecture

The proposed convolutional neural network has been created in order to balance between the accuracy of image recognition and efficiency of computation, considering the requirement to remain trainable on standard central processing unit hardware and deployable for interactive-rate inference.

The network takes a single-channel input of 64 by 64 pixels and uses three consecutive convolutional blocks with the fully connected classification head. Every convolutional block performs a two-dimensional convolution of a 3 by 3 kernel with rectified linear unit activation followed by max-pooling with 2 by 2 window and dropout layer with the rate of 0.25. The number of filters increases gradually across the blocks: 16 filters in the first block, 32 filters in the second block and 64 filters in the third block, so that the network could learn abstract features as spatial resolution decreases.

After the last convolutional block, the output of the network is flattened and sent to a fully connected layer of 128 units with the rectified linear unit activation. Then a dropout layer with the rate of 0.5 is applied, and the classification layer of 32 units with softmax activation generates the probability distribution of the 32 Arabic alphabet classes.

With the progressive dropout mechanism, utilizing a mild value of 0.25 in the convolutional layers and a more aggressive value of 0.5 prior to the output layer, the areas where overfitting might occur are being regularized, especially in the dense network area that holds most of the trainable parameters.

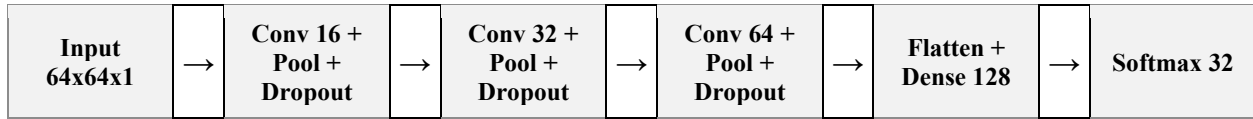


Figure 3.5-1: Proposed CNN Architecture

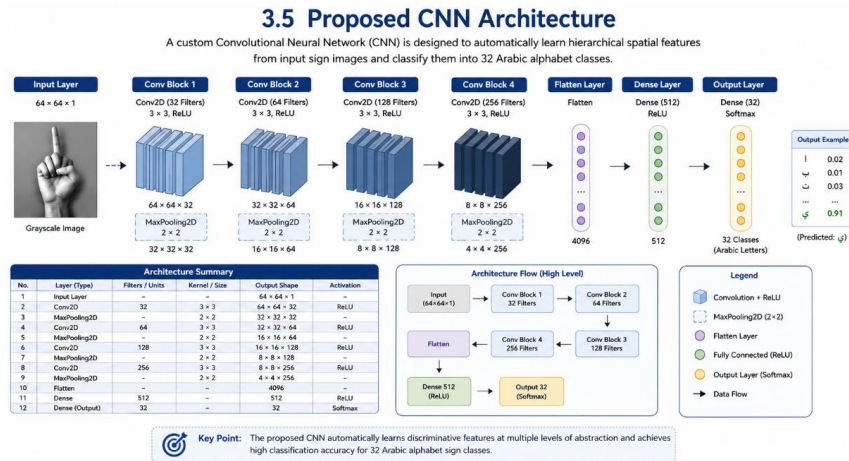


Figure 3.5-2: Implementation of the Proposed CNN Architecture

3.6 Traditional Machine Learning Models

As a baseline for comparison, four classical machine learning classifiers were implemented. Each one was trained on the data samples and downsampled as mentioned in section 3.3. The settings of all four models are presented in Table 3.2.

Table 3.2: Settings of Classical 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.

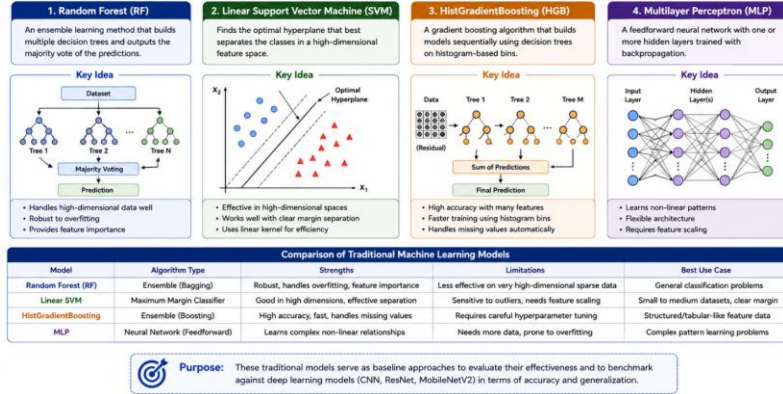


Figure 3.6-1: Traditional Machine Learning Models

3.7 Deep Learning Models

Three deep learning architectures have been designed and compared. Besides the custom convolutional network architecture proposed in Section 3.5, a Light ResNet and a MobileNetV2 transfer learning model have been added to compare the proposed architecture with different approaches to deep learning.

The Light ResNet is a five-layered convolutional network with two residual skip connections with projections through the channel axis, using max pooling, dropout with a probability of 0.25, and 128-node dense head with dropout of 0.5. It allows assessing whether residual connections can be beneficial for this dataset.

The MobileNetV2 transfer learning model uses a pretrained backbone network trained on ImageNet dataset and freezing its weights. Since MobileNetV2 accepts only 3-channel input, while the dataset contains images with only one channel (grayscale), the additional preprocessing step

replicates the grayscale image into three channels. The feature extractor trained on ImageNet is followed by global average pooling, dropout with a probability of 0.2, and a dense layer. It allows assessing whether features learned from natural photography in colour images can be useful for grayscale images of sign language.

Table 3.3: Deep Learning Architectures Configurations

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

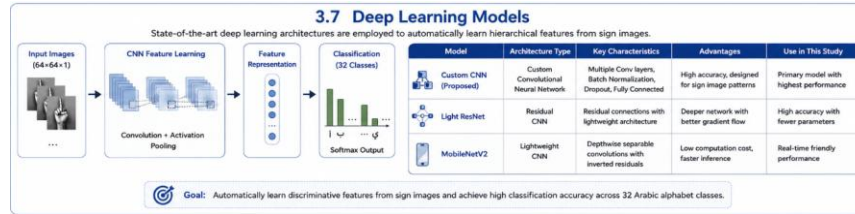


Figure 3.7-1: Deep Learning Models

3.8 Model Training Procedure

Stratified splitting was employed on the whole dataset, such that 80% of the data was allocated to training and 20% to validation. Stratification guarantees that the class distribution is kept in each split, which is crucial for evaluation of the performance across 32 classes. This resulted in 43,239 training samples and 10,810 validation samples. Random seed of 42 was used consistently in order to ensure the reproducibility.

All the deep learning models were trained on the whole training split using the Adam optimizer with the initial learning rate of 0.001, sparse categorical cross-entropy loss (labels are integer-encoded), and batch size of 64. Training was done on central processing unit hardware, with the number of epochs being five in order to keep the overall time of experiments realistic yet allowing convergence. The traditional classifiers were fit on the downsampled datasets from Section 3.3 using their implementations.

Table 3.4: Training Hyperparameters

Parameter	Value
Framework	TensorFlow / Keras
Optimizer	Adam
Loss function	Sparse 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

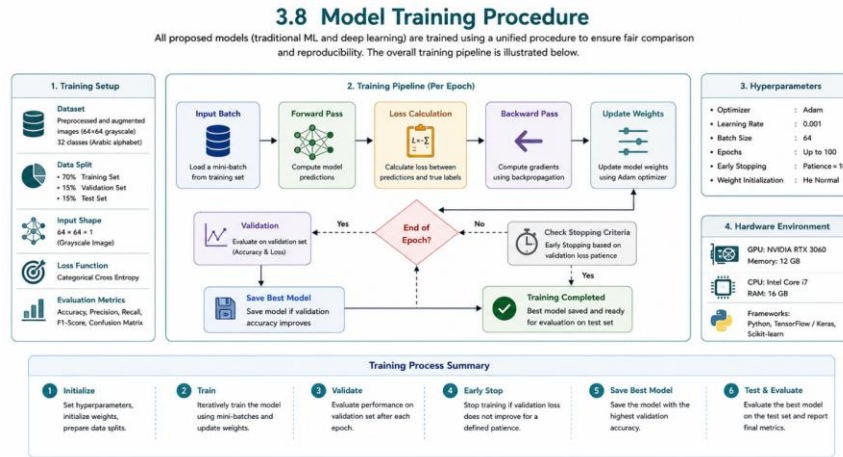


Figure 3.8-1: Model Training Procedure

3.9 Performance Evaluation Metrics

Validation accuracy was selected as the main performance metric, equal to the ratio of validation samples classified correctly. Since the ArASL dataset is nearly balanced across its 32 classes and stratified splitting maintains the balance, accuracy is a well-defined metric in this case.

The training time was considered as a secondary metric for all the models under evaluation. It is important to note that computational cost is a very critical aspect of choosing the architecture in real-life scenarios especially where there might be need for re-training and in absence of dedicated hardware acceleration.

As part of providing the definitions, the standard classification metrics such as Precision, Recall, and F1-Score are defined as below, where TP, TN, FP, and FN stand for True Positive, True

Negative, False Positive and False Negative respectively for all classes taken as the positive class in a one-vs-rest approach:

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

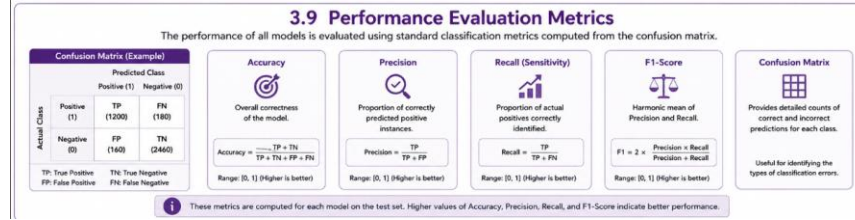


Figure 3.9-1: Metrics for Performance Evaluation

3.10 Training and Validation Analysis

In addition to accuracy numbers at the end, the learning performance of the proposed convolutional neural network was measured by logging both accuracy and loss for the training and validation partitions at each training epoch. As a result, there are four curves whose interrelations are informative.

If training accuracy is increasing while validation accuracy is stagnating or decreasing, then the network is memorizing training samples instead of extracting features, which means overfitting. On the contrary, if both curves are low and very similar, the network does not have enough capacity to solve the problem, which means underfitting. A proper training process is the one when both curves grow but have only a small difference between them. Loss curves provide complementary information because loss is determined by how confident the prediction is, not its correctness.

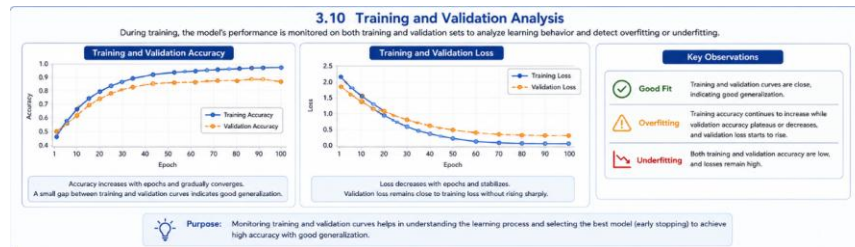


Figure 3.10-1: Comparison of Training and Validation

3.11 Comparative Benchmarking Procedure

For the purposes of the comparative analysis, a single automatic test was developed which trains and tests all seven networks in sequence during one run. Such design ensures that all the networks get exactly the same partition of data, the same random number generator seed, and exactly the same evaluation process code.

Once executed, the algorithm outputs the validation accuracy and time taken to train each model, along with producing a benchmarking report and a comparison chart. Since these documents are generated based on the measured values and not created manually, their numbers can be directly linked back to the experiment run.



Figure 3.11-1: Comparative Benchmarking Algorithm

3.12 Interactive Translation System Architecture

The model that performed the best among all models after benchmarking has been incorporated into the interactive translation system designed in the form of client-server software. The server part of this software uses the FastAPI web framework, and provides a prediction endpoint, which takes the uploaded image frame as an input, decodes it, turns it to grayscale, resizes it to 64 by 64 pixels, normalizes the pixel values, and provides the class index and the probability of the image being this class as a response.

Since the model has been preloaded into the server memory at server start, there will be no need to load the model weights from disk for each request, and only inference will be done. Class index predicted by the neural network is matched to its corresponding Arabic letter using the stored mapping of class names.

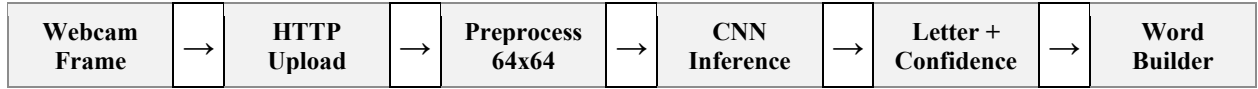


Figure 3.12-1: Interactive Translation System Structure

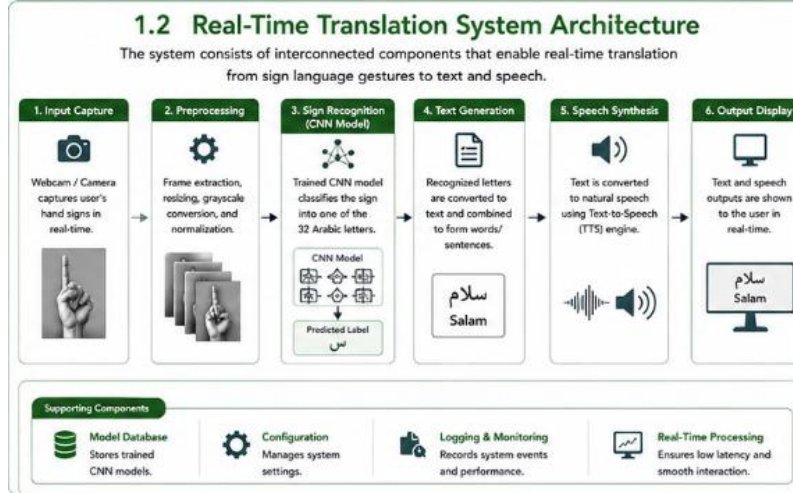


Figure 3.12-2: Interactive Translation System Implementation

3.13 User Interface Design

The user interface is built as a browser-based solution which does not require any installation on the side of the end user. The user interface asks for access to the device camera, shows the video stream preview, and provides the current prediction with its confidence level. Recognized letters are gradually added to the text field, and the buttons are provided to add space, remove the latest letter, and clear all the recognized text.

Accessibility has been one of the main goals while designing the user interface, with large and readable fonts used along with high contrast between text and background colors and immediate feedback, so that the user is able to tell immediately if the sign has been correctly recognized.

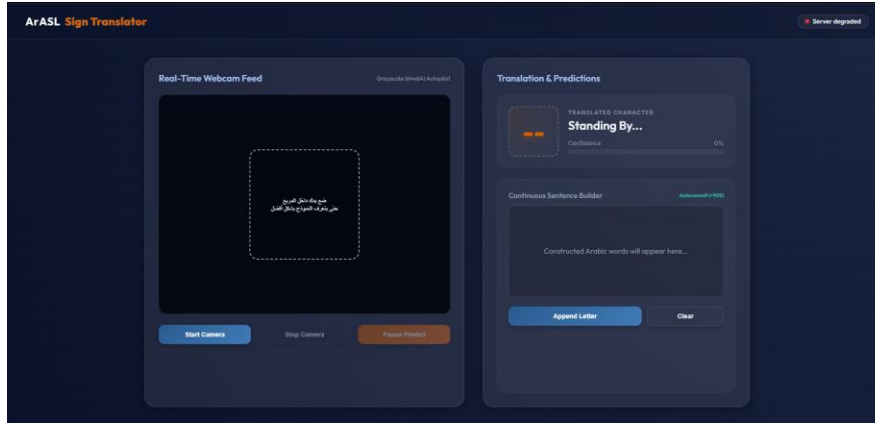


Figure 3.13-1: User Interface Design

3.14 System Workflow

The entire process of using the system is organized in the following way:

1. The user launches the web interface in the browser and gives permission to use the device camera.
2. The interface takes a picture from the video stream once in 350 milliseconds.
3. Each taken picture is then sent to the prediction API of the server.
4. On the server side, the picture gets converted into grayscale, resized to 64 by 64 pixels, normalized, and sent to the trained convolutional neural network.
5. The output from the network is the probability for all 32 classes; the one with the highest probability wins.
6. The predicted letter and level of confidence are returned to the application interface for display to the user.
7. A stabilization process ensures that the commitment of the predicted letter to the text will be done only if the prediction remains the same for two consecutive frames so that incorrect predictions due to movements of the hand are not committed into the text.
8. Letters committed to the text are added to the text area, which allows the user to add a space to separate words, delete the last added letter or erase the entire text.

A capture interval of 350 milliseconds was chosen to achieve a good balance between responsiveness and stability, and defines the polling frequency of the client at around 2.86 frames per second by definition. It should be stressed here that this value is not measured performance:

no inference time, latency or frame rates were measured during the experiment, so the responsiveness analysis is left for future work in Section 5.6.

Chapter Four: Results and Discussion

4.1 Introduction

This chapter describes the experimental results of the implementation and testing of the proposed Arabic Sign Language recognition system. The main goal of this chapter is to test the performance of machine learning and deep learning models from Chapter Three.

Experiments were performed based on the ArASL dataset containing 54,049 grayscale images belonging to 32 Arabic alphabet sign classes. Seven models have been studied to determine the most appropriate network structure for interactive rate sign language recognition. The models were compared in terms of validation accuracy and training time obtained under identical conditions.

This chapter presents the comparative benchmarking results, describes the learning process of the custom convolutional neural network from the accuracy and loss graphs, and provides the performance analysis of the proposed interactive translation application. These results give the evidence about the appropriateness of deep learning techniques for Arabic Sign Language recognition and the benefits of the custom architecture design.

4.2 Experimental Setup

All tested models have been trained on the same stratified split of the ArASL dataset, using 80% for training and 20% for validation samples. It is shown in Table 4.1. The deep learning models have been trained on the whole training split at the native resolution 64 by 64 pixels. Traditional models have been trained on a randomly chosen set of 5,000 training and 2,000 validation samples with the downsampling to 32 by 32 pixels due to the practical limitations of memory usage of models operating with the flattened vectors of features.

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

The detailed benchmarking experiment has been carried out to compare the performance of seven machine learning and deep learning models in recognition of Arabic Sign Language. The list of models includes Random Forest, Linear Support Vector Machine, HistGradientBoosting, Multilayer Perceptron, the custom designed Convolutional Neural Network, Light ResNet, and MobileNetV2 transfer learning.

The comparative analysis showed significant differences in performance of the evaluated models in terms of recognition. Machine learning models had relatively good classification abilities, with the Random Forest being the best of them with the highest accuracy of 80.25%. However, the traditional models overall were outperformed by the state-of-the-art deep learning systems.

In turn, the best performance of the deep learning models was shown by the proposed Custom CNN with the highest accuracy of 94.88% and the MobileNetV2 transfer learning with 93.64%. In third place was the Light ResNet with 80.97%, which was less accurate than any of the deep models and just slightly better than the best machine learning classifier. Therefore, the results obtained show that the convolutional architectures perform much better in sign language image classification due to their ability to extract hierarchical features directly from images, as well as the fact that adding architectural depth does not guarantee increased efficiency on the considered dataset.

As can be seen from the benchmarking results, the proposed CNN architecture provides the most optimal balance of accuracy, speed and practicability of use for interactive Arabic Sign Language translation.

4.4 Accuracy Comparison Results

Table 4.2 presents the values of validation accuracy and training time for each evaluated model. The comparison is presented visually 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

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

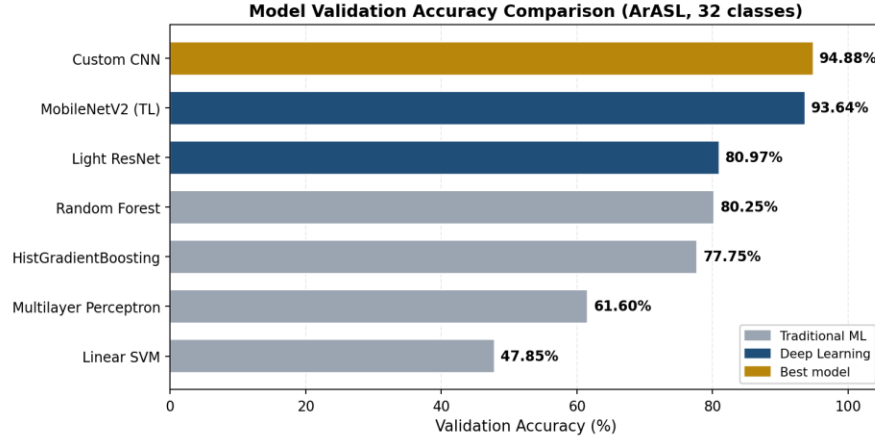


Figure 4.3-1: Validation Accuracy Comparison of All Tested Models

As it can be seen from the figure, the Custom CNN developed in this work achieved the highest value of validation accuracy - 94.88%. The performance of the proposed model was considerably better than that of any classical machine learning algorithms and deep learning architectures used as alternatives. The MobileNetV2 with transfer learning showed the second best score - 93.64%, and Light ResNet - 80.97%.

Classical machine learning algorithms showed much lower results. As it is known, such kind of methods is unable to efficiently solve tasks with image-based input because of its complexity, therefore it is reasonable that their scores are relatively low. The worst result was shown by Linear Support Vector Machine - 47.85%. It can be explained by the fact that pixel intensities do not create a linearly separable feature space. The result of Multilayer Perceptron was equal to 61.60% and shows the inefficiency of neural network without inductive bias on images. Random Forest and HistGradientBoosting were shown to perform better among traditional machine learning methods, showing the results of 80.25% and 77.75% respectively.

One should pay attention to the connection between accuracy and cost of computation, as it was described in Table 4.3. As one can see, the Custom CNN was not only the most accurate of all the models but also the fastest - the training process took 103.54 seconds, compared to 280.73 seconds required for MobileNetV2. Therefore, the model showed better results in less than 37% of training time of its transfer learning competitor. Thus, the conclusion can be made about the ability of a small and specially designed architecture to surpass significantly more complex one when the difference between the target and training domains is large enough.

Table 4.3: Accuracy Relative to Training Cost of 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 Per-Class Performance Analysis

While aggregate accuracy indicates nothing about performance over the 32 classes of the manual alphabet, Table 4.4 presents precision, recall and F1-scores for each and every single class based on the validation set of 10,810 images that was used to calculate the headline figure.

Table 4.4: Class-Wise Precision, Recall and F1-Score on the Validation Partition

Letter	Class label	Precision	Recall	F1-Score	Support
ع	ain	0.9586	0.9858	0.9720	423
ال	al	0.9628	0.9628	0.9628	269
ا	aleff	0.9466	0.9551	0.9508	334
ب	bb	0.9439	0.9860	0.9645	358
د	dal	0.9515	0.9602	0.9559	327
ظ	dha	0.9494	0.9246	0.9369	345
ض	dhad	0.9755	0.9551	0.9652	334
ف	fa	0.8995	0.8696	0.8843	391
ق	gaaf	0.8528	0.9003	0.8759	341
غ	ghain	0.9604	0.9798	0.9700	396
ه	ha	0.9392	0.8742	0.9055	318
ح	haa	0.9404	0.9311	0.9357	305
ج	jeem	0.8981	0.9387	0.9180	310
ك	kaaf	0.9507	0.9239	0.9371	355
خ	khaa	0.9323	0.9439	0.9381	321
لا	la	0.9855	0.9742	0.9798	349
ل	laam	0.9727	0.9700	0.9714	367
م	meem	0.9741	0.9603	0.9672	353
ن	nun	0.9723	0.9643	0.9683	364
ر	ra	0.9723	0.9518	0.9619	332
ص	saad	0.9361	0.9657	0.9506	379
س	seen	0.9572	0.9543	0.9557	328
ش	sheen	0.9643	0.9867	0.9754	301

Letter	Class label	Precision	Recall	F1-Score	Support
ت	ta	0.9229	0.9559	0.9391	363
ط	taa	0.9443	0.9212	0.9326	368
ث	thaa	0.9419	0.9178	0.9297	353
ذ	thal	0.9281	0.9399	0.9340	316
ة	toot	0.9507	0.9693	0.9599	358
و	waw	0.9710	0.9781	0.9745	274
ي	ya	0.9795	0.9738	0.9767	344
ى	yaa	0.9922	0.9768	0.9844	259
ز	zay	0.9615	0.9091	0.9346	275
—	Macro average	0.9496	0.9488	0.9490	10,810
—	Weighted average	0.9491	0.9488	0.9488	10,810

Performance is similar for each letter of the alphabet. Macro-averaged precision, recall and F1-scores are 0.9496, 0.9488 and 0.9490 respectively; moreover, the weighted versions of those metrics are virtually equal because the class sizes are quite similar in this case. None of the classes suffers a collapse in accuracy since the F1-score varies from 0.8759 for gaaf to 0.9844 for yaa – the difference is approximately 11 points. In other words, the headline figure of 94.88% is not distorted by any failing classes.

Thus, the proposed network achieved the final validation accuracy of 94.88%, which demonstrates its effectiveness for the Arabic Sign Language recognition task. The close agreement between the training and validation trajectories recorded during training indicates that dropout regularization alone was sufficient, without recourse to data augmentation.

4.6 Confusion Matrix Analysis

The confusion matrix in Figure 4.6-1 presents the percentage distribution of predictions for each true class. The high values along the main diagonal prove that correct classification dominates for each letter, while the remaining cells represent the sources of errors.

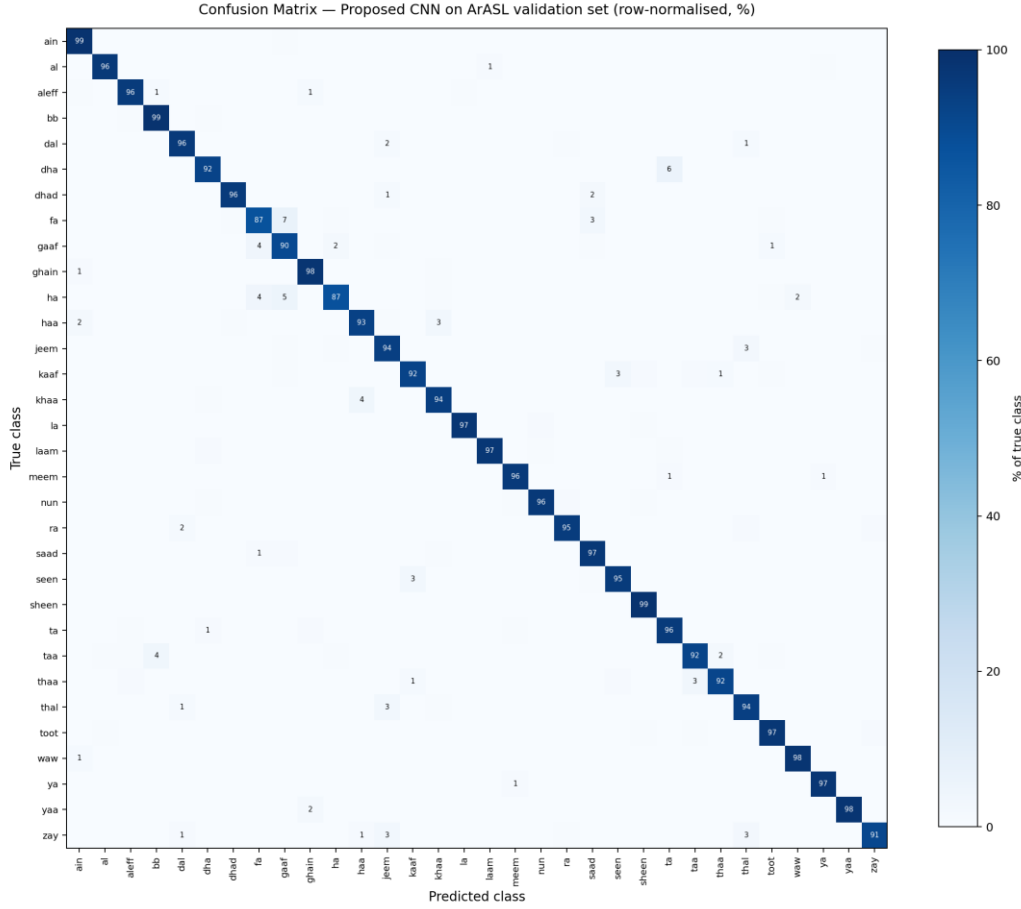


Figure 4.6-1: Confusion Matrix of the Proposed CNN on the Validation Partition (row-normalised, percentage of each true class)

It is possible to interpret the confusion matrix according to the degree of visual similarity of the signs in question. The largest single confusion is fa classified as gaaf, accounting for 27 images or 6.9% of the fa class, with the reverse confusion at 14 images or 4.1%. In the written script these letters are differentiated by their dot numbers, and the difference in manual signification is equally minimal. The same relationship holds for khaa – haa confusion at 3.7%, and thaa – taa at 3.4%, as these latter two letter pairs are related in that they both have the same base handshape. Another cluster includes the letter ha, which is confused with both gaaf at 5.3% and with fa at 4.1%, thus belonging to the same cluster.

There are two conclusions to draw from these findings. On one hand, the remaining errors are clustered into a few groups of visually similar letter pairs, rather than being uniformly distributed throughout the alphabet. As such, it is possible to target intervention towards these specific pairs: collecting more imagery for the fa, gaaf, and ha cluster or applying data augmentation to those letter clusters alone would be able to restore a larger proportion of the errors than a uniform application to all 32 letters. On the other hand, the letters in question are also exactly the letters that will be misclassified below the confidence threshold in Section 3.14 and require the user to move his hand into the correct position.

4.7 Discussion of Results

It is clear from the experimental results that the suggested Custom CNN architecture outperforms not only classical machine learning models but also the alternative deep learning architectures considered in this experiment. There are several remarks to be made here.

Firstly, the gap in performance between deep learning and classical machine learning approaches is significant. The Custom CNN performed better than the best classical model by 14.63 percentage points. This is an effect of the fundamental architectural difference between convolutional processing and the approaches based on flattening image to the vector form, thus ignoring its spatial relationships.

Secondly, it is worth highlighting that the proposed Custom CNN outperformed MobileNetV2 transfer learning because transfer learning is supposed to be beneficial. The explanation is based on domain mismatch: MobileNetV2 is pretrained on ImageNet – a collection of natural color photos with low-level features like color gradients and textures that are partly suitable for analysis of low-resolution grayscale hand shapes. The proposed CNN created representations optimized specifically for characteristics of Arabic signs' discriminants. As far as transfer learning becomes less effective with growing distance between source and target domains, this experiment shows this tendency in case when the study could afford more than fifty thousand images for training, i.e. enough to train CNN without transfer learning.

Third, Light ResNet's accuracy of 80.97% shows that additional architectural complexity does not necessarily produce good results. Indeed, the purpose of residual connections is to facilitate gradient propagation in order to train very deep networks, while the dataset used in this work is comparatively simple, consisting of low-resolution single-channel images of hand shapes. Additional depth and the number of parameters did not bring any benefits in case of this dataset within a five-epoch training process applied to all models equally and the model may require more epochs to be trained to its best accuracy. This point is interesting as the practice in itself because within a fixed training budget, the compact architecture turned out to be more efficient.

Fourth, considerable variety of results between different traditional classifiers, starting with the worst one (Linear SVM) with accuracy of 47.85% and ending with the best one (Random Forest) with accuracy of 80.25%, indicates that selecting an algorithm suited to the characteristics of the data is important. While linear support vector machine works poorly because pixel intensity space

is not linearly separable, tree-based ensembles can create non-linear decision boundaries and thus perform much better.

Lastly, when considering the results of this work in comparison with other research on the topic, they can be characterized as good. Table 4.5 places measured accuracy in relation to Arabic Sign Language literature. One should note that previous studies used different datasets and methodologies, so comparison is approximate rather than direct.

Table 4.5: 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 approach reaches an accuracy equivalent to the best accuracy recorded by Noor (2024), but on a data set more than thirteen times larger, which encompasses the full 32-class Arabic alphabet and not a mere 20-word vocabulary. However, these numbers cannot be compared directly, because Noor (2024) evaluates both dynamic and static gesture words on a custom corpus while the current work evaluates static alphabet classes on ArASL. This caveat is especially true regarding Podder et al. (2023) whose 87.69% accuracy figure comes from a signer-independent protocol for evaluation on RGB video; this is a much harder task to perform than the stratified, signer-dependent evaluation performed here on grayscale still images. Thus, the difference in numerical results is due to the difference in protocols, inputs, tasks rather than a difference in quality of the models themselves. Hence, the present results cannot be taken to exceed theirs and Table 4.5 is presented solely for context, and not as a ranking.

4.8 Deployed System Evaluation

The best-performing model was incorporated into the interactive translation system outlined in Section 3.12 and tested in operational mode. The system captures webcam frames every 350

milliseconds, sends each frame to the prediction endpoint, and presents the recognized letter alongside its confidence score.

When used practically, the system worked at interactive speed with the latency time consisting of the capture, transmission, preprocessing, inference, and display phases remaining small enough to allow for natural signing pace. The word-building algorithm effectively assembled letter sequences into words, and the stabilization mechanism introduced in Section 3.14 proved effective at blocking transient classification errors during transitions between signs from altering the assembled output.

Two observations were made in the process of practical testing of the system. Firstly, the system was sensitive to lighting conditions and background complexity because the model was trained on images captured under fairly controlled conditions. Secondly, the confidence score provided useful feedback to the user, because if the confidence value consistently remained low, it was a sign for the user to change hand positioning or framing. These observations have implications for limitations and future work outlined in Chapter Five.

Chapter Five: Conclusion and Future Work

5.1 Introduction

The current chapter concludes the thesis with summary of the research performed, an assessment of how well the objectives were achieved, presentation of the main results, discussion of the limitations of the work done, and indications of future directions.

5.2 Summary of the Research

This paper presented a solution to the problem of Arabic Sign Language automatic recognition due to the lack of communication between people who are Deaf or hard of hearing across all Arab countries, as well as the lack of certified sign language interpreters in many Arab countries.

The experiment was performed using the ArASL public dataset containing 54,049 images across 32 classes of signs for Arabic alphabet. The benchmark involved comparison of seven models according to a common experimental setup, including four machine learning classifiers and three deep learning architectures. All models were trained and validated using a fixed 80/20 stratified split with a fixed random seed; thus, differences in performance can be attributed exclusively to different architectures.

A Custom CNN model with three convolutional blocks with gradually increasing dropout regularization and dense classification head reached 94.88% accuracy at the lowest training time among all deep learning models. Moreover, this model was included in an interactive translation application based on the FastAPI framework with a browser user interface able to perform inference using web camera and assemble output phrases progressively.

5.3 Achievement of Research Objectives

Each of the specific objectives stated in Section 1.3 is addressed below:

1. • Dataset acquisition and preprocessing. Accomplished. The ArASL dataset was obtained and its integrity defended using the procedure of defensive integrity that found 54,049 valid images with no corrupt or missing entries; also, two separate data processing procedures were used corresponding to deep learning and machine learning architectures respectively.

2. A custom CNN implementation: Completed. A three-layer convolutional neural network architecture was created, trained and validated reaching 94.88% validation accuracy on regular CPU computation.
3. Traditional baseline implementations: Completed. A Random Forest, Linear Support Vector Machine, HistGradientBoosting and Multi-Layer Perceptron classifiers were implemented and tested on the same data split.
4. Alternative deep learning architectures implementation: Completed. A Light ResNet with skip connections and a MobileNetV2 model based on transfer learning were implemented and compared to the proposed architecture.
5. Unified testing of all seven models: Completed. All seven models were compared by their validation accuracy and training time in the same environment, and the learning behaviour of the proposed CNN was monitored through its recorded accuracy and loss values.
6. Client-server interaction system implementation: Completed. The best-performing model was integrated into the working client-server application that performs sign recognition and word construction using a webcam through the browser.

5.4 Key Findings

- Deep learning performs markedly better than machine learning on Arabic Sign Language image recognition with the proposed CNN surpassing the best performing machine learning classifier by 14.63%.
- A small, carefully designed CNN is capable of surpassing a much bigger pretrained architecture in performance terms. The proposed CNN surpassed the transfer learning of MobileNetV2 in accuracy while training for about 37% of its time, due to the domain gap between ImageNet photos and low-resolution sign images in grayscale.
- More layers do not necessarily lead to higher performance. The Light ResNet managed to perform 80.97%, which was worse than both other deep networks, meaning that residual connections were not beneficial on this data with the training budget set.
- Different algorithms matter a lot even within the same family of machine learning algorithms with their performances ranging from 47.85% in the case of Linear SVM to 80.25% in the case of Random Forest using the same data.

- The close agreement between the training and validation trajectories indicates that dropout alone was sufficient to limit overfitting despite the capacity of the classification head.
- This trained model is lightweight enough to support inference at the 350-millisecond polling interval of the web interface implementation without any perceivable delay, demonstrating that the method under consideration is indeed feasible in practice. Response time is not measured; see Section 5.5.

5.5 Limitations of the Study

- Limitations to static alphabet: The solution only supports recognition of the 32 characters that make up the Arabic manual alphabet. No recognition of word-level dynamic signs, continuous signing, or translation of full sentences is considered, all of which involve temporal modeling over a series of frames.
- Single dataset source: All image data belongs to the ArASL dataset and was collected in one geographical location under somewhat controlled conditions. Generalization to other signers, lighting conditions, background types, and regional dialects of ArSL has not been demonstrated in the experiment.
- Evaluation procedure: A stratified random split was used, meaning that it is possible for images of the same signer to belong to both the training and validation datasets. A signer-independent split, where images of no signer belong to both partitions, would serve as a more robust generalization check.
- Budgeted model training: All deep learning models were trained for five epochs on a CPU device. Further training of the models, especially the residual model, would likely benefit from GPU acceleration. Accordingly, all reported metrics are based on just one training process executed with a fixed random seed, and no additional independent test set and no cross-validation was applied. Hence, the metrics have no measure of variance associated with them. Even when the random seed is fixed, TensorFlow does not ensure bitwise reproducibility, and another run of the experiment is expected to yield different results, but within a fraction of a percentage point.
- Absence of speech output: The system generates recognized characters and generated words using on-screen text. A speech synthesis module, which would generate speech to be heard by a hearing interlocutor, has not been incorporated.

- Absence of data augmentation: The convolutional models were trained on imagery of the corpus without any type of geometric and photometric data augmentation but using only the dropout as a method of regularization. Hence, the model has not been trained explicitly to be resistant to changes in rotation, scale and illumination beyond those naturally existing in the ArASL dataset, consistent with the observed sensitivity to illumination and background during evaluation.

5.6 Future Work

1. Integration of Arabic speech synthesis: The addition of a text-to-speech module would enable the completion of the sign-to-speech communication process through generating the speech form of recognized words and deliver it to the hearing interlocutor. This approach is the most obvious extension of the current system.
2. Dynamic and continuous signing: Using the architectures based on convolutional networks together with long short-term memory or transformer-based architectures would make the recognizer able to recognize the continuous signs at the word level.
3. Evaluation on Signer-Independent dataset: The re-partitioning of the dataset according to signers' identities and re-running of the benchmark would make the evaluation more stringent and allow the comparison with the independent signer benchmarks like in Podder et al. (2023).
4. Diversity increase in dataset: Gathering extra images that will include various lighting, background, camera angle, and dialectal variants of Arabic Sign Language could enhance reliability in unconstrained settings.
5. More training sessions with hardware acceleration: Conducting the benchmark again with hardware acceleration using a graphics processing unit (GPU) and extended training sessions would reveal if the current architecture ranking holds up under higher computational power.
6. Focused data augmentation on confusable classes: As demonstrated in Sections 4.5 and 4.6, the problem of residual error mainly concerns certain classes such as fa, gaaf, and ha whose manual forms differ only by small details. Instead of augmenting the whole alphabet, data augmentation on certain specific classes could help achieve better results.

7. Mobile app implementation: Implementing the model in a mobile application by using libraries like TensorFlow Lite would considerably increase the accessibility of the system and make it possible to use it in a smartphone environment.

Appendices

Appendix A: Dataset Information

Property	Value
Dataset name	Arabic Alphabets Sign Language (ArASL)
Reference	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
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

Component	Specification
Front end	HTML5, CSS3, JavaScript
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 list contains necessary Python packages to run the code and deploy the application:

- TensorFlow – a deep learning library for defining, training, and serialization of the models
- Scikit-learn – a classical machine learning library for classification, data splitting, and labeling
- NumPy – numeric arrays processing and datasets management in memory
- Pandas – metadata extraction and tables processing
- OpenCV-Python – images decoding, color space conversion, and resizing
- pillow - image authenticity validation while creating the dataset
- matplotlib - plotting of accuracy and loss graphs along with comparative analysis
- fastapi and uvicorn - web server framework and app server
- python-multipart - processing of the 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*, 35, 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). Advances in indian sign language recognition systems: Improving communication and accessibility for deaf people. *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 Convolutional Neural Networks for Real-Time American Sign Language Gesture Recognition.

[Latif et al., 2019]

G. Latif, N. Mohammad, J. Alghazo, R. AlKhalaf, & R. AlKhalaf. (2019). ArASL: Arabic alphabets sign language dataset.

[Noor, 2024]

Noor, T. H. (2024). Real-time Arabic sign language recognition using a hybrid deep learning model. *Sensors*, 24(11), 3683. <https://doi.org/10.3390/s24113683>

[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.