



# **Arabic Alphabet Sign Language Translator**

Prepared By

Name: Yasir Aljarroof      ID No.: 243001813

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

At

Collage of Informatics

Midocean University

UAE

Supervisor Name: Ashraf Elsayed

Collage of Informatics, Midocean University, Comoros

Email: 243001813@st.midocean.edu.km

## **Abstract:**

This thesis presents the development of AI-powered applications for translating Arabic Alphabet Sign Language into text, addressing the communication challenges faced by the deaf community. The project focuses on creating a tool that can recognize and translate static sign language gestures accurately into Arabic text which will help bridge the communication barriers using deep learning models. The methodology involves collecting a dataset of Arabic Alphabet sign language images and implementing hand detection and gesture recognition to finally translate these into text. The developed model can recognize 32 of the Arabic sign Language including the 28 alphabet letters plus 4 more gestures. A hypered model was developed with a CNN pre-trained model (EfficientNetB0) that used the actual image to classify it on one end, and a fully connected model that uses landmarks points extracted from the image (extracted with mediapipe). The hypered model achieves high accuracy reaching 98%.

## **Acknowledgements**

I would like to thank Dr. Ashraf Elsayed for the support and continuous guidance throughout this research project that led me to these results.

# **Table of Contents**

## **CHAPTER 1: INTRODUCTION**

- 1.1 Background And Motivation
- 1.2 Problem Statement
- 1.3 Research Objectives

## **CHAPTER 2: LITERATURE REVIEW**

- 2.1 Introduction
- 2.2 Related Work

## **CHAPTER 3: DATASET**

- 3.1 Dataset Description
- 3.2 Preprocessing

## **CHAPTER 4: PREPOSED APPROACH**

- 4.1 Framework
- 4.2 Models architecture
- 4.3 Data Preprocessing And Augmentation
- 4.4 Hyperparameter Tuning

## **CHAPTER 5: EXPERIMENTAL RESULTS AND DISCUSSION**

- 5.1 Resnet50 Model
- 5.2 EfficientNetB0 Model
- 5.3 Dense Neural Network Model
- 5.4 The Hybrid Model

## **CHAPTER 6: CONCLUSION AND FUTURE WORK**

### 6.1 Conclusion

### 6.2 Future Work

# CHAPTER 1: INTRODUCTION

## 1.1 Background and motivation

The deaf community faces significant communication barriers in their daily interactions with hearing individuals. This tool will help the deaf community so that they don't have to rely on sign language interpreters to assist with communication which presents substantial challenges like the availability of sign language interpreters and the lack of automated translation tools. This project addresses this critical gap by using artificial intelligence and computer vision technologies to train a model for Arabic alphabet sign recognition. The goal is to help deaf people communicate better and feel included while also improving AI for Arabic Alphabet sign language recognition.

## 1.2 Problem Statement

The current lack of Arabic Alphabet Sign Language knowledge creates significant barriers for deaf individuals trying to communicate with hearing people. The challenge lies in developing an automated translation system that can accurately recognize and translate Arabic Alphabet Signs while being accessible, user-friendly, and practical for everyday use.

## 1.3 Research objectives

The research aims to develop an AI-powered system for accurate Arabic Alphabet Sign Language recognition and translation and design a user-friendly interface, achieving high accuracy in sign recognition while maintaining real-time processing capabilities.

## CHAPTER 2: LITERATURE REVIEW

### 2.1 Introduction

The advancements in deep learning have made huge progress, which helped in a lot of areas. One of these areas is the sign language recognition system, especially for Arabic and Saudi Sign Language , which is aiming to bridge communication gaps between deaf and hearing communities. Researchers have mostly used convolutional neural networks (CNNs) for static gesture classification, though hybrid models and real-world applications remain areas of active exploration.

### 2.2 Related work

Al-Obodi et al. (2020) [1] made a CNN-based system for Saudi Sign Language recognition, achieving high accuracy by training on 40 classes of static gestures, including 28 alphabetical signs, numbers 1-10 signs, and two more gestures for me “أنا” and “لا”. Their work demonstrated the viability of CNNs for isolated gestures. However, it had limitations including a restricted vocabulary and reliance on still images rather than continuous sentences. Similarly, El Rwelli et al. (2021) [4] developed a CNN-driven system to translate Arabic Sign Language (ArSL) gestures into vocalized speech, attaining 90% accuracy. However, their model’s training on only 30 sign letters highlighted a recurring challenge in sign language recognition: limited vocabulary scalability.

To address dynamic gestures, hybrid architectures combining CNNs with sequential models have emerged. Noor et al. (2024) [3] proposed a CNN-LSTM framework for real-time ArSL recognition. They have achieved 94.40% accuracy for static gestures, and 82.70% for dynamic ones. They have trained the models on 10 static words and 10 dynamic gestures. The results were great, but the dataset was limited, suggesting the need for larger datasets for both static and dynamic gestures. Al Khuzayem et al. (2024) [2] have developed a CNN-based SSL recognition model and made it into a mobile application, Efhamni, which supports both static and dynamic gestures. Despite its innovation, the system required further testing in varied lighting and background conditions, as the data set is not

available to see, and the sample provided had different people doing the signs, but all had the same lighting and background.

The efficacy of CNNs for static gestures is further supported by Samonte et al. (2022) [5] , whose research and literature review of 20 sign language recognition studies have highlighted CNNs and Connectionist Temporal Classification (CTC) as dominant for static and dynamic recognition.

All together, these studies highlight CNN models robustness in static gesture recognition but reveal some challenges. like limited vocabulary, dynamic gesture recognition less than static performance. Future work must prioritize larger, diverse datasets, and different environments.

## CHAPTER 3: DATASET

### 3.1 Dataset Description

The dataset [6] we chose for this project is Arabic Sign Language Letters Dataset (ArSL21L) . . Its is made of 14,202 Images of 32 distinct Arabic sign language letter signs, including 28 arabic letters, and the following:” ال, لا, ة, ئ”. This data set has various backgrounds, and was collected from 50 different subjects. The data set has been evaluated using the YOLOv5l model achieving high results.

## 3.2 Preprocessing

Before I start training the models, the first step was to check the dataset and make sure it has enough images for each class, and that it's balanced between them. The next step was to resize the images to a consistent input size (224x224) to be able to train the model faster without losing details. Next was splitting the data to 70% training, 20% testing, and 10% validation. Then we proceed to do Data Augmentation by applying rotations, horizontal flips, slight shifts, and slight zoom to the training images.

This was the initial data preprocessing. Later we also used a Mediapipe by Google to first identify where the hands are in the images, and crop the image to remove unwanted information which helped the model to achieve better results. But after this step, we needed to check, validate and remove unwanted images from the data set as some images had a second hand (resting hand for example) in them which the Mediapipe also cropped and added to the dataset. Also, for each of these images made by Mediapipe, I saved a text file that contained the landmarks that were identified by Mediapipe for each image (21 landmark \* 3 coordinates each for a total of 63 features), to be used later to train a model and test the results.

## CHAPTER 4: PROPOSED APPROACH

### 4.1 Framework

First, the frameworks. I chose Keras with TensorFlow as my deep learning framework. It's a very powerful framework. I chose it for its ease of use and it can build models with just a few lines of code. Although it required more effort to have it set up and working with GPU on windows.

### 4.2 Models architecture

Since CNN seems to be the most popular choice and has a high accuracy from previous studies, I chose to explore and test out some CNN pretrained models to learn distinctive features present in the dataset. We also explored using Dense Neural Networks with the extracted landmarks.

The final model architecture for the CNN was: EfficientNetB0 base\_model, GlobalAveragePooling2D, two dense layers (512 and 256), each followed by a dropout of 0.3, then an output layer with softmax activation function with the 32 classes.

As for the Dense Neural Networks, it was made out of 3 dense layers (256-128-64), each followed by a dropout layer (0.3-0.2-0.1)

### 4.3 Data Preprocessing And Augmentation

In this step, we first used mediapipe by google to crop the images around the hands, and also saved a copy of the hands landmarks to use with our model. Then we have resized the images to a consistent input size (224x224). Next we did Data Augmentation, applying rotations, horizontal flips, slight shifts, and slight zoom to images, and normalized the landmarks using StandardScaler in order to balance the dataset for better generalization.

## 4.4 Hyperparameter Tuning

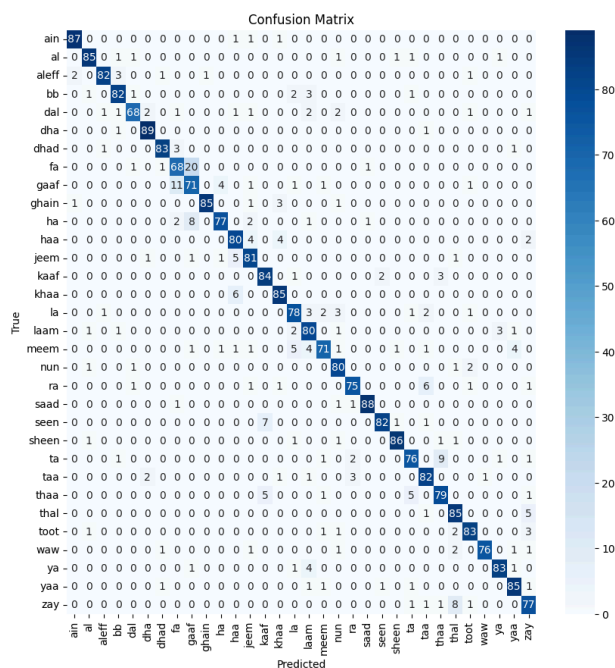
Experiment with the number of dense layers, learning rates, dropout rates, and unfreezing layers from the base model and compared the results to find the optimal result for this.

## 5.1 Resnet50 Model

First, we used the dataset as it is since it had almost the same number of images across classes. The first model we tried was ResNet50, with base\_model, GlobalAveragePooling2D, Dense layer (128), dropout of 0.5, then an output layer with softmax activation function with the 32 classes. Then we started fitting the model with a 0.0001 learning rate. After 50 epochs, the validation accuracy reached 57% and the model stopped improving.

For the second try, I changed the model architecture. I tried ResNet50, with `base_model`, `GlobalAveragePooling2D`, two dense layers (512 and 256), each followed by a dropout of 0.3, then an output layer with softmax activation function with the 32 classes. After 115 epochs, it reached 64% validation accuracy then stopped improving. Then I tried unfreezing the layers of resnet50 to make the model learn and adopt more from our arabic sign language dataset with lower learning rate (0.00001). This has improved the model drastically, achieving 89% validation accuracy after 40 more epochs.

Accuracy: 0.8974537844436693  
Precision: 0.8999788239166187  
Recall: 0.8974537844436693  
F1 Score: 0.8978338469210082



## 5.2 EfficientNetB0 Model

### Trial 1:

The next model we tried was EfficientNet B0 with the same set up as our last model. This was able to get the same results in less epochs. And after unfreezing 30% of the layers, it reached 91% validation accuracy.

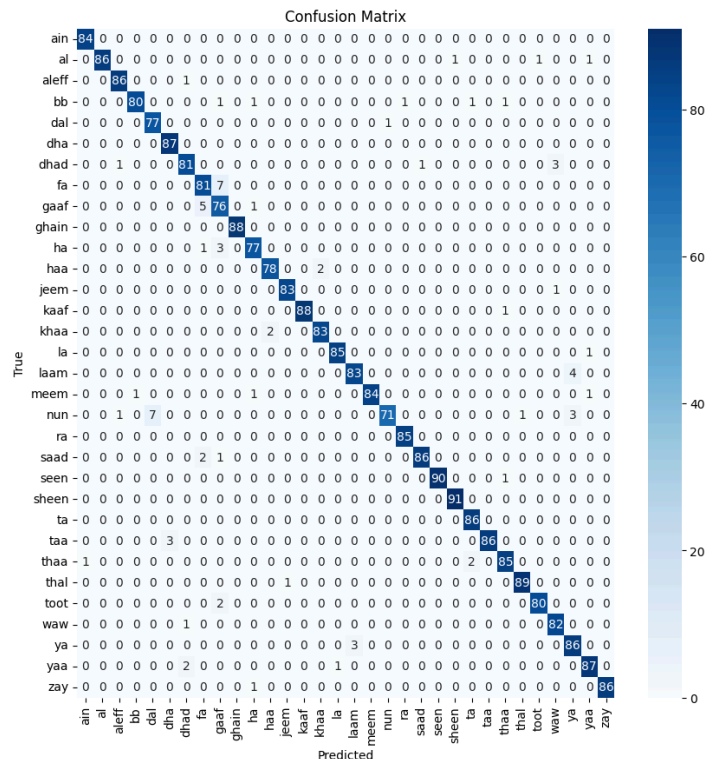
### Trial 2:

After getting to 91% validation accuracy, the model struggled to improve. So I was looking for other way to increase the model capacity to learn from the dataset. So I started to do more preprocessing of the dataset to try to achieve better results. For this, I used mediapipe by google to detect the hand, and crop the image around it to make the image more focused on the hand and ignore the rest. With the new dataset (cropped images around hands) with EfficientNet B0, achieving 83% validation accuracy before unfreezing any layers. Then we proceeded with unfreezing layers little by little until we reached 97% validation accuracy.

## Evaluations:

**Accuracy: 0.9713352685050798**

**Precision: 0.972138468091689**

**Recall: 0.9713352685050798****F1 Score: 0.9714005632523358**

## 5.3 Dense Neural Network Model

The next step was to try another approach to this. When I used mediapipe, i also saved the landmarks of the detected hand (21 landmark \* 3 coordinate each). So I used these as my dataset and fed them to a Dense Neural Network with 3 dense layers (256-128-64), each followed by a dropout layer (0.3-0.2-0.1), then an output layer with softmax activation function with the 32 classes. This was the fastest model to train since it's not actually an image, and the fastest to get great results reaching around 90% at the 7th epoch. At the end of the training, the model reached 96% validation accuracy which is very close to our results with EfficientNetB0.

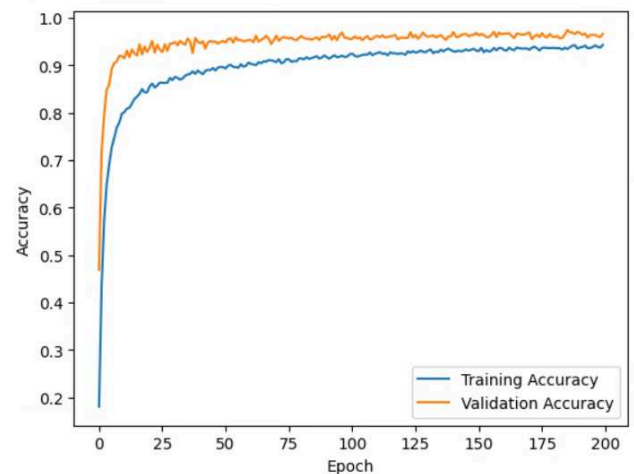
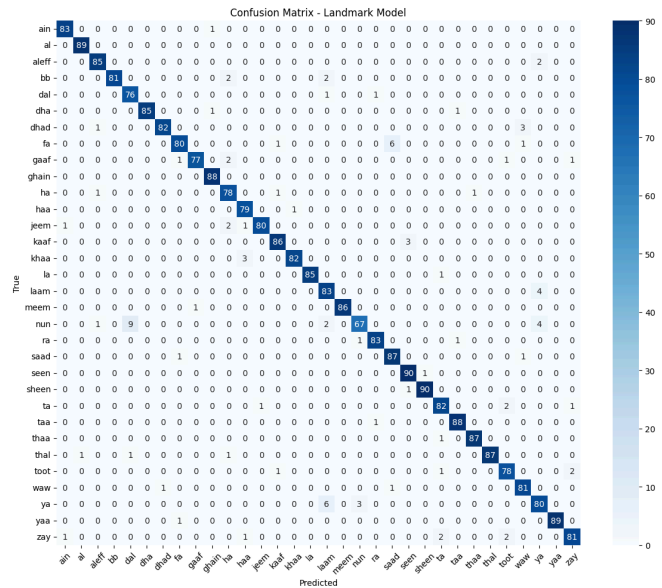
### Evaluations:

**Accuracy: 0.9633526850507983**

**Precision: 0.964171288683201**

**Recall: 0.9633526850507983**

**F1 Score: 0.9633067003743021**



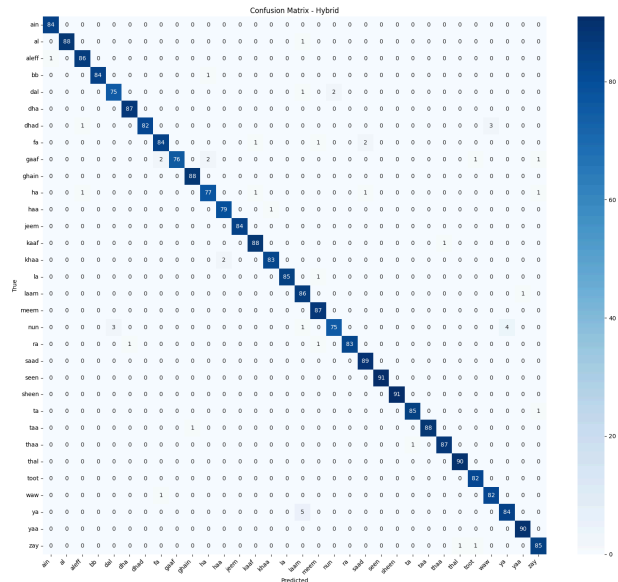
## 5.4 The Hybrid Model

While analyzing the result for both of these two models, I noticed that each model excelled in different parts. For example, the EfficientNetB0 model had people distinguishing between “taa” and “dha”, but is better distinguishing between “kaaf” and “seen”. So I made a decision tree method that will take the prediction of both of these models, and decide which one to pick based on the two predictions and the confidence of the predictions. By doing so, the final Hyperd model have reached an accuracy of 98%. The results show that combining both approaches provides better accuracy compared to using each model individually. After analyzing the wrong prediction, I can see that its because the gestures are a lot similar, sometimes a slight tilt changes the sign letter.

## Evaluations:

**Accuracy: 0.9815****Precision: 0.9818****Recall: 0.9815****F1 Score: 0.9814**

| Per-Class Analysis: |       |        |        |            |
|---------------------|-------|--------|--------|------------|
| Class               | Total | EF_Acc | LM_Acc | Hybrid_Acc |
| ain                 | 84    | 1.000  | 0.988  | 1.000      |
| al                  | 89    | 0.978  | 1.000  | 0.989      |
| aleff               | 87    | 0.966  | 0.977  | 0.989      |
| bb                  | 85    | 0.965  | 0.976  | 0.988      |
| dal                 | 78    | 1.000  | 0.949  | 0.962      |
| dha                 | 87    | 1.000  | 0.989  | 1.000      |
| dhad                | 86    | 0.988  | 0.965  | 0.953      |
| fa                  | 88    | 0.943  | 0.943  | 0.955      |
| gaaf                | 82    | 0.902  | 0.927  | 0.927      |
| ghain               | 88    | 1.000  | 1.000  | 1.000      |
| ha                  | 81    | 0.926  | 0.926  | 0.951      |
| haa                 | 80    | 0.975  | 0.988  | 0.988      |
| jeem                | 84    | 0.988  | 0.964  | 1.000      |
| kaaf                | 89    | 0.966  | 0.955  | 0.989      |
| khaa                | 85    | 0.976  | 0.976  | 0.976      |
| la                  | 86    | 0.988  | 0.988  | 0.988      |
| laam                | 87    | 0.954  | 0.943  | 0.989      |
| meem                | 87    | 0.989  | 0.989  | 1.000      |
| nun                 | 83    | 0.855  | 0.892  | 0.904      |
| ra                  | 85    | 0.976  | 0.976  | 0.976      |
| saad                | 89    | 1.000  | 0.989  | 1.000      |
| seen                | 91    | 1.000  | 0.989  | 1.000      |
| sheen               | 91    | 1.000  | 1.000  | 1.000      |
| ta                  | 86    | 1.000  | 0.965  | 0.988      |
| taa                 | 89    | 0.966  | 0.989  | 0.989      |
| thaa                | 88    | 0.977  | 0.989  | 0.989      |
| thal                | 90    | 0.978  | 0.967  | 1.000      |
| toot                | 82    | 0.988  | 0.951  | 1.000      |
| waw                 | 83    | 0.988  | 0.964  | 0.988      |
| ya                  | 89    | 0.966  | 0.910  | 0.944      |
| yaa                 | 90    | 0.967  | 0.989  | 1.000      |
| zay                 | 87    | 0.966  | 0.897  | 0.977      |



## **CHAPTER 6: CONCLUSION AND FUTURE WORK**

### **6.1 Conclusion**

Both EfficientNetB0 and Dense Neural Network models have achieved a great result individually, with the EfficientNetB0 performing slightly better by about 1%. Combining both have achieved a 98% accuracy.

### **6.2 Future Work**

The next step is to Extend the system to recognize full words and continuous gestures. Then, integrate and deploy it to a mobile application to help bridging the communication gap between deaf and hearing people and reduce the challenges faced by the deaf community.

## References

1. Al-Obodi, A. H., Al-Hanine, A. M., Al-Harbi, K. N., Al-Dawas, M. S., & Al-Shargabi, A. A. (2020). A Saudi Sign Language Recognition System based on Convolutional Neural Networks. *Building Services Engineering Research and Technology*, 13(11), 3328–3334. <https://doi.org/10.37624/IJERT/13.11.2020.3328-3334>
2. Al Khuzayem, L., Shafi, S., Aljahdali, S., Alkhamesie, R., & Alzamzami, O. (2024). Efhamni: A Deep Learning-Based Saudi Sign Language Recognition Application. *Sensors*, 24(3112). <https://doi.org/10.3390/s24103112>
3. Noor, T. H., Noor, A., Alharbi, A. F., Faisal, A., Alrashidi, R., Alsaedi, A. S., Alharbi, G., Alsanoosy, T., & Alsaedi, A. (2024). Real-Time Arabic Sign Language Recognition Using a Hybrid Deep Learning Model. *Sensors*, 24(3683). <https://doi.org/10.3390/s24113683>
4. El Rwelli, R., Shahin, O. R., & Taloba, A. I. (2021). Gesture-Based Arabic Sign Language Recognition for Impaired People Using CNN. *International Journal of Advanced Computer Science and Applications*, 12(12), 574–582. <https://doi.org/10.14569/IJACSA.2021.0121273>
5. Samonte, M. J. C., Guingab, C. J. M., Relayo, R. A., Sheng, M. J. C., & Tamayo, J. R. D. (2022). Using Deep Learning in Sign Language Translation to Text. *Proceedings of the International Conference on Industrial Engineering and Operations Management*, Istanbul, Turkey, 4036–4044.

6. Munkhjargal Gochoo (2022). Arabic Sign Language Dataset 2022. Kaggle. Retrieved from:  
<https://www.kaggle.com/datasets/ammarsayedtaha/arabic-sign-language-dataset-2022>