



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 my parents for all their love and support and encouragement that helped me throughout this journey.

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

List of Abbreviations.....	5
List of Figures.....	6

CHAPTER 1: INTRODUCTION	7
1.1 Background And Motivation	7
1.2 Problem Statement	7
1.3 Research Objectives	7
CHAPTER 2: LITERATURE REVIEW	8
2.1 Introduction	8
2.2 Related Work	8
CHAPTER 3: DATASET	
3.1 Dataset Description	10
3.2 Preprocessing	10
CHAPTER 4: PREPOSED APPROACH	11
4.1 Framework	11
4.2 Models architecture	11
4.3 Data Preprocessing And Augmentation	12
4.4 Hyperparameter Tuning	12
CHAPTER 5: EXPERIMENTAL RESULTS AND DISCUSSION	13

5.1 Resnet50 Model	13
5.2 EfficientNetB0 Model	15
5.3 Dense Neural Network Model	19
5.4 The Hybrid Model	22
CHAPTER 6: CONCLUSION AND FUTURE WORK	25
6.1 Conclusion	25
6.2 Future Work2	25

List of Abbreviations

- **Acc.:** Accuracy
- **CNN:** Convolutional Neural Network
- **ArSL:** Arabic Sign Language
- **LSTM:** Long Short-Term Memory
- **SSL:** Secure Sockets Layer
-
- **CTC:** Connectionist Temporal Classification

List of Figures

5.1 Resnet50 Evaluation & Confusion Matrix	14
5.2 Similar Sign Gestures Examples	16
5.3 EfficantNetB0 Evaluation & Confusion Matrix	18
5.4 Dense Neural Network Evaluation & Confusion Matrix	20
5.5 Dense Neural Network Training And Validation Accuracy Plot	21
5.6 Hybrid Model Architecture	23
5.7 Hybrid Model Evaluation & Confusion Matrix	24

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.

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: PREPOSED APPROACH

4.1 Framework

The framework used in this experiment was Keras with TensorFlow as the deep learning framework. It's a very powerful framework that was chosen for its ease of use and readability, 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 Pre trained 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)

After exploring different models, I went with a hybrid approach, combining two models to get the best results out of them. The hybrid approach is made from CNN EfficientNetB0 that takes the image as input, and a Dense Neural Network Model that takes landmarks extracted for that image using MediaPipe as input.

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.

CHAPTER 5: EXPERIMENTAL RESULTS AND DISCUSSION

5.1 Resnet50 Model

Trial 1:

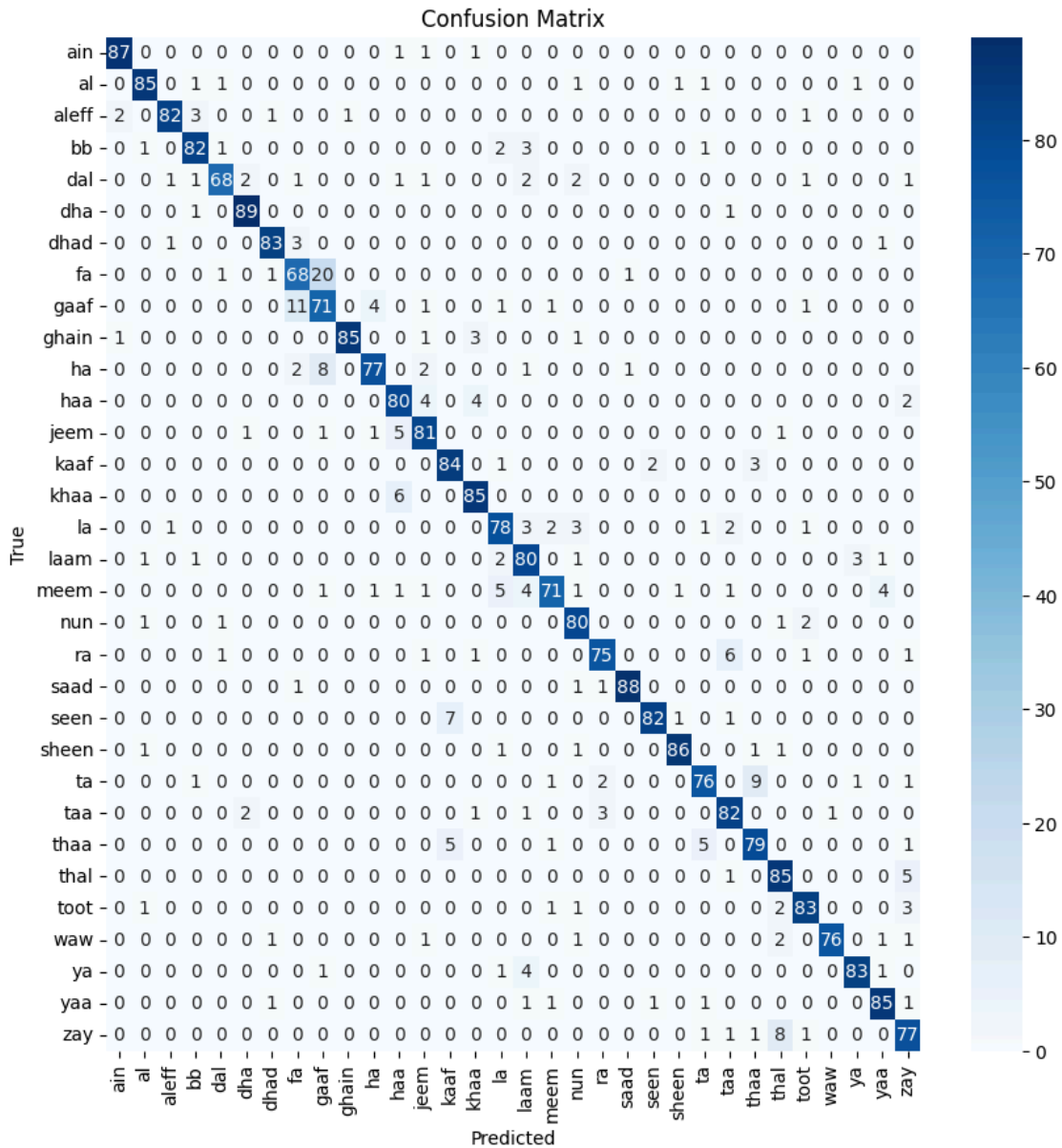
First, we used the dataset as it is since it had almost the same number of images across classes. The first model we tried was ResNet50, with base_model, GlobalAveragePooling2D, Dense layer (128), dropout of 0.5, then an output layer with softmax activation function with the 32 classes. Then we started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

Trial 2:

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. Model evaluation and confusion matrix are shown in figure 5.1.1.

Figure 5.1 - Resnet50 Evaluation & Confusion Matrix

Accuracy	Precision	Recall	F1 Score
0.8974	0.8999	0.8974	0.8978



5.2 EfficientNetB0 Model

Trial 1:

The next model we tried was EfficientNet B0 with the same set up as our last model. 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. This was able to get the same results in less time and epochs. But after around 50 epochs, the validation accuracy stopped improving, and started fluctuating while the accuracy was still increasing. That was the first sign of over fitting. So the next step was loading the best model before the over fitting started. Then, I proceeded to unfreeze 30% of the layers to give the model more room to learn and tune to the dataset. Doing so has helped the model and produced very good improvements, helping it reach 91% validation accuracy. But it couldn't get higher than this. The model was unable to learn more from the dataset after this. It was struggling with some of the signs that had close shapes, like Jeem and Haa, Dal and Nun, etc. see figure 5.2.1 for some examples the model struggled with.

Figure 5.2 - Similar Sign Gestures Examples

Jeem



Haa



Dal



Nun



Trial 2:

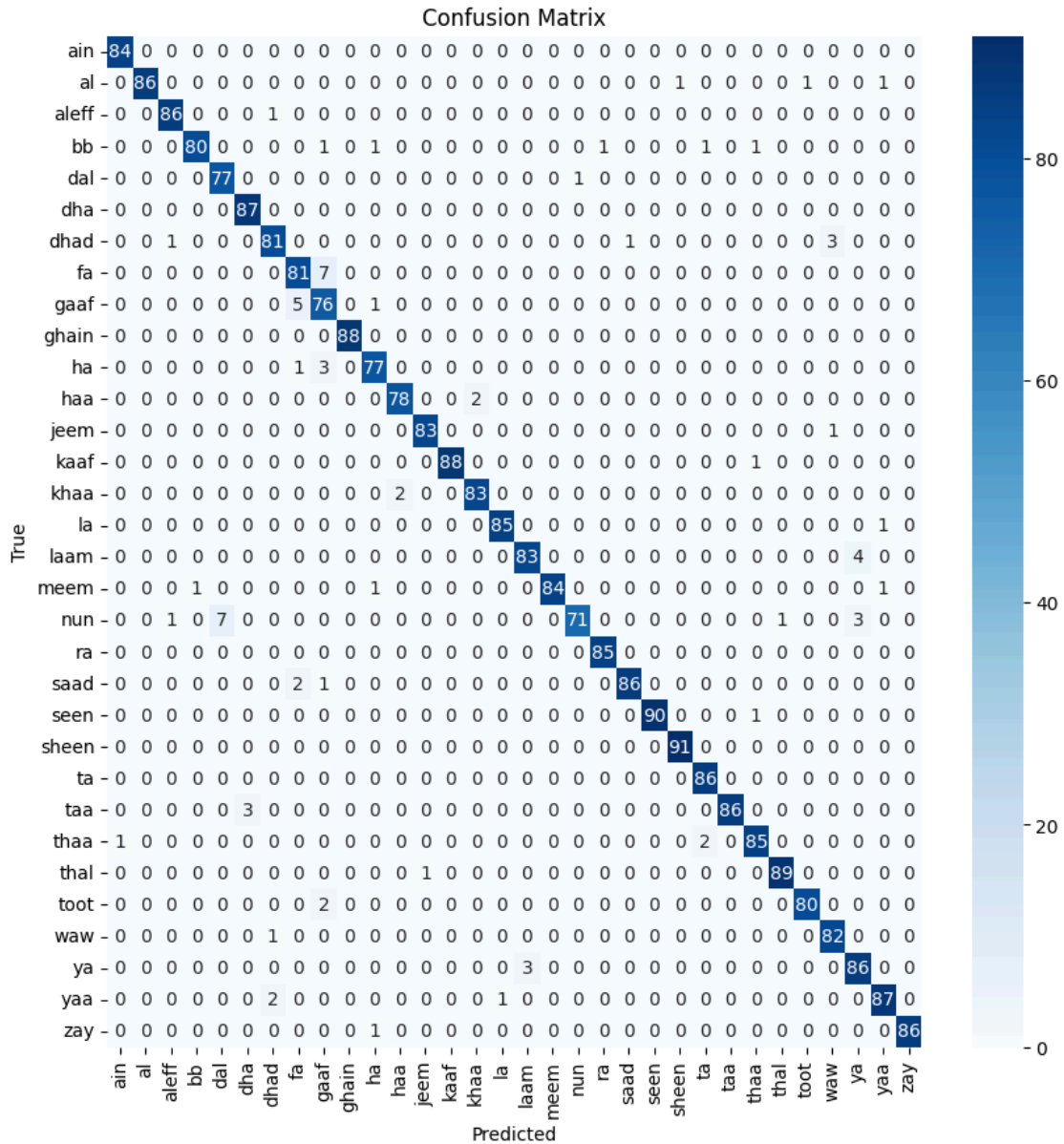
After getting to 91% validation accuracy, the model stopped improving and started to overfit. So I tried changing the learning rate, using a learning rate scheduler, and changing the batch size, but all of these had little to no effect. So instead of trying to change the model, I started to do more preprocessing of the dataset to try to achieve better results. After deep analyzing the dataset I discovered that even though all the images had the same size, the hands in the images were at different depth from the camera, which meant that some images were focused on the hand, and some had lots of background space. So the idea was to crop the image around the hand and resize all to the same size. The idea is great but it's not practical to go through 14000 images and crop the images one by one. After doing some research on AI tools for hand detection, I decided to use mediapipe by google to first detect the hand, then crop the image around its coordinates to make the image more focused on the hand and ignore the rest.

This did not mean that the data set is now perfect, as some images had two hands (one doing the sign and the other was just relaxed). At first I used the dataset as it is but the results were good for some classes, but too random for the others. So I did more preprocessing by actually looking through all the images. This is when I found the issue. After deleting all the images of the wrong hands (around 5000 images) the dataset was ready for testing with our model.

With the new dataset with the EfficientNet B0, the model was able to achieve 83% validation accuracy in under 20 epochs and before unfreezing any layers. Once the model stopped improving, we proceeded with unfreezing layers little by little until we reached 97% validation accuracy. As shown in figure 5.2.2, the classes that had the most mis predictions are Nun and Dal as shown in Figure 5.2.1 previously.

Figure 5.3 - EfficantNetB0 Evaluation & Confusion Matrix

Accuracy	Precision	Recall	F1 Score
0.9713	0.9721	0.9713	0.9714



5.3 Dense Neural Network Model

The next step was to try another approach to this. Since we already used a tool to crop the images around the hands by detecting the hand landmarks, we saved these landmarks (21 landmarks * 3 coordinate each for a total of 63 features) and used this as our dataset to train a model. These were split the same way across train (70%), test (20%), and validate (10%) and used to train 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.

The results of this approach were marvelous timewise. 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 which only took 3 mins to complete, compared to efficientB0 which took around an hour and 30 mins to reach the same results. At the end of the training, the model reached 96% validation accuracy which is very close to our results with EfficientNetB0.

Figure 5.4 - Dense Neural Network Evaluation & Confusion Matrix

Accuracy	Precision	Recall	F1 Score
0.9633	0.9641	0.9633	0.9633

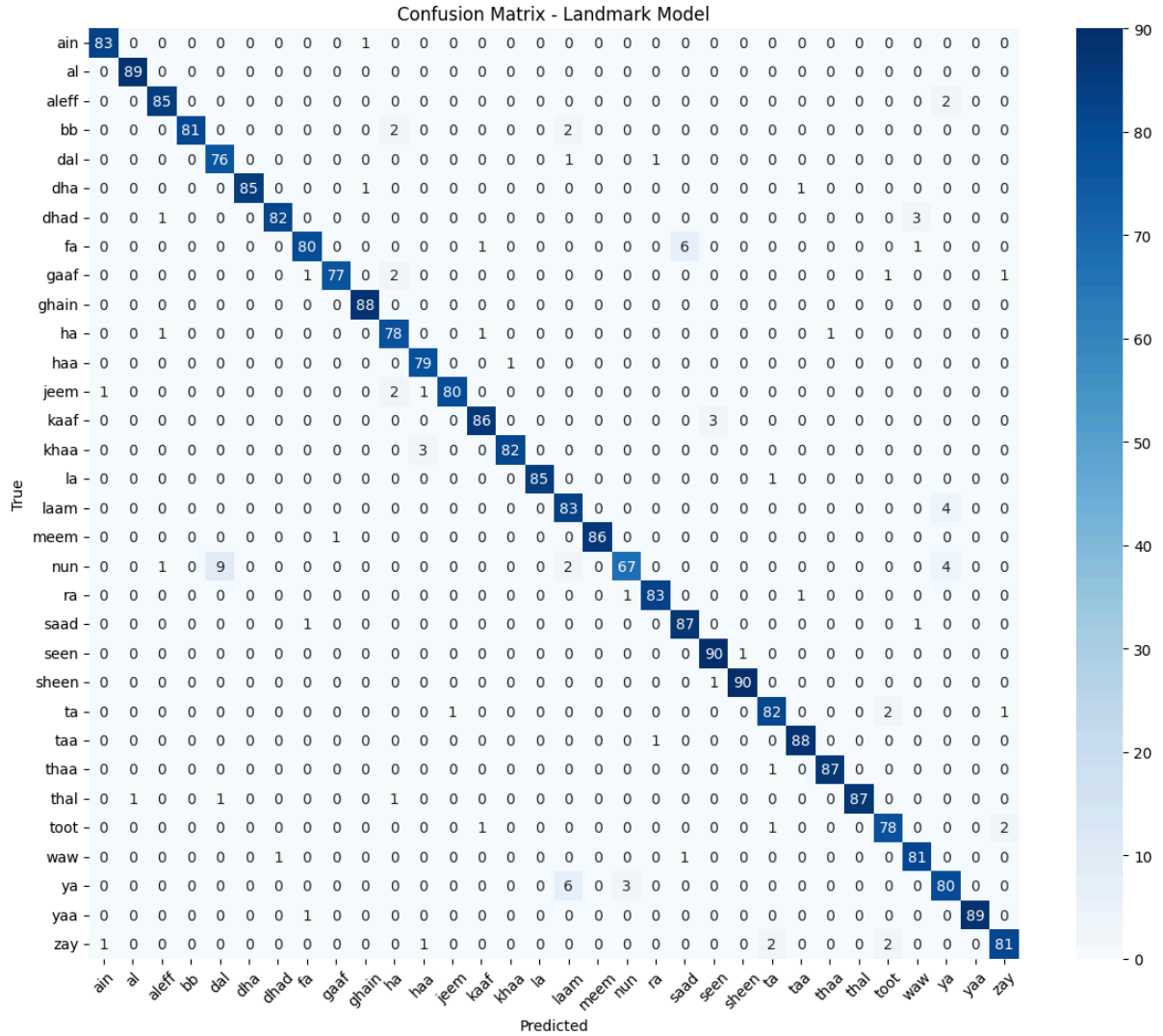
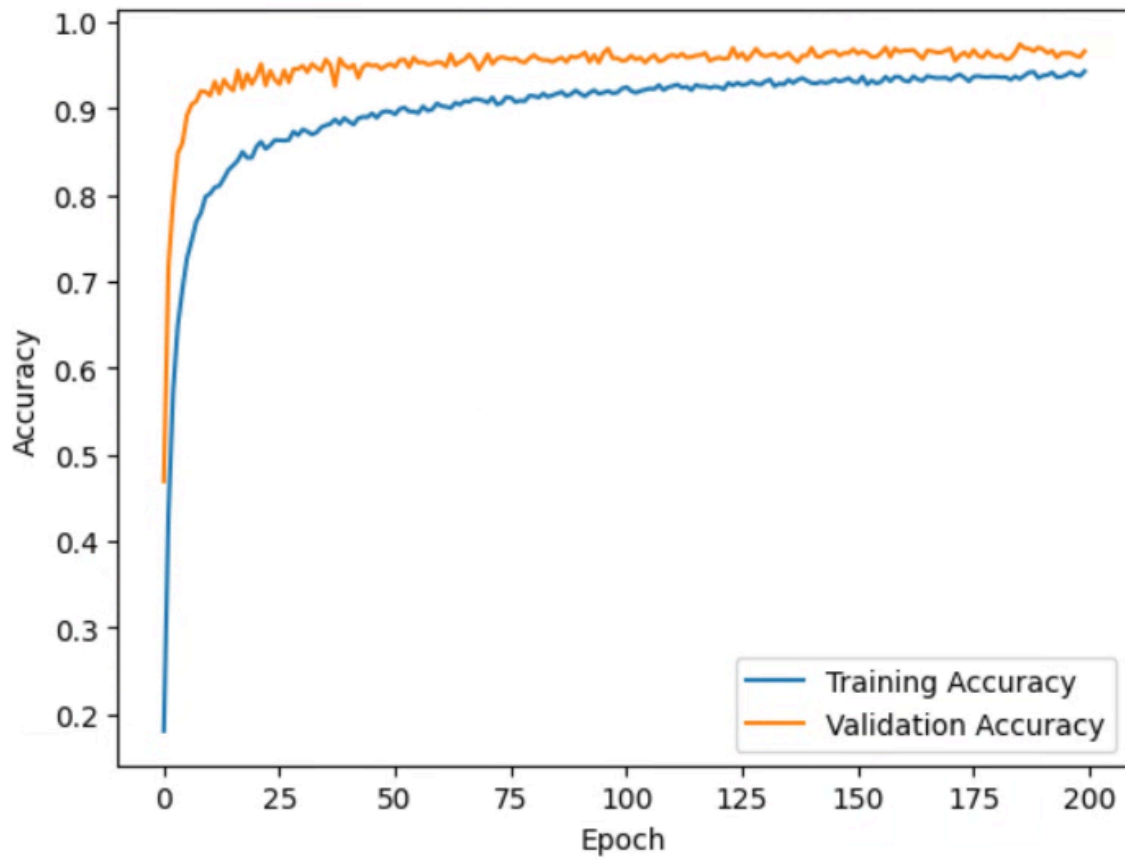


Figure 5.5 - Dense Neural Network Training And Validation Accuracy accuracy



5.4 The Hybrid Model

While analyzing the result for both of these two models, I noticed that each model excelled in predicting different classes. For example, the EfficientNetB0 model had people distinguishing between “taa” and “dha”, but is better distinguishing between “kaaf” and “seen”. By analyzing both results of these two models, I have developed a method that will take the prediction, confidence of each, and decide which prediction to use. This method takes these inputs, and passes them to a decision tree that was made from analyzing the results and confusion matrix of both models. By doing so, the final Hybrid model has reached an accuracy of 98%, combining the best out of both models to achieve a higher accuracy rate. 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 it's because the gestures are a lot similar, sometimes a slight tilt changes the sign letter from one to another.

Figure 5.6 - Hybrid Model Architecture

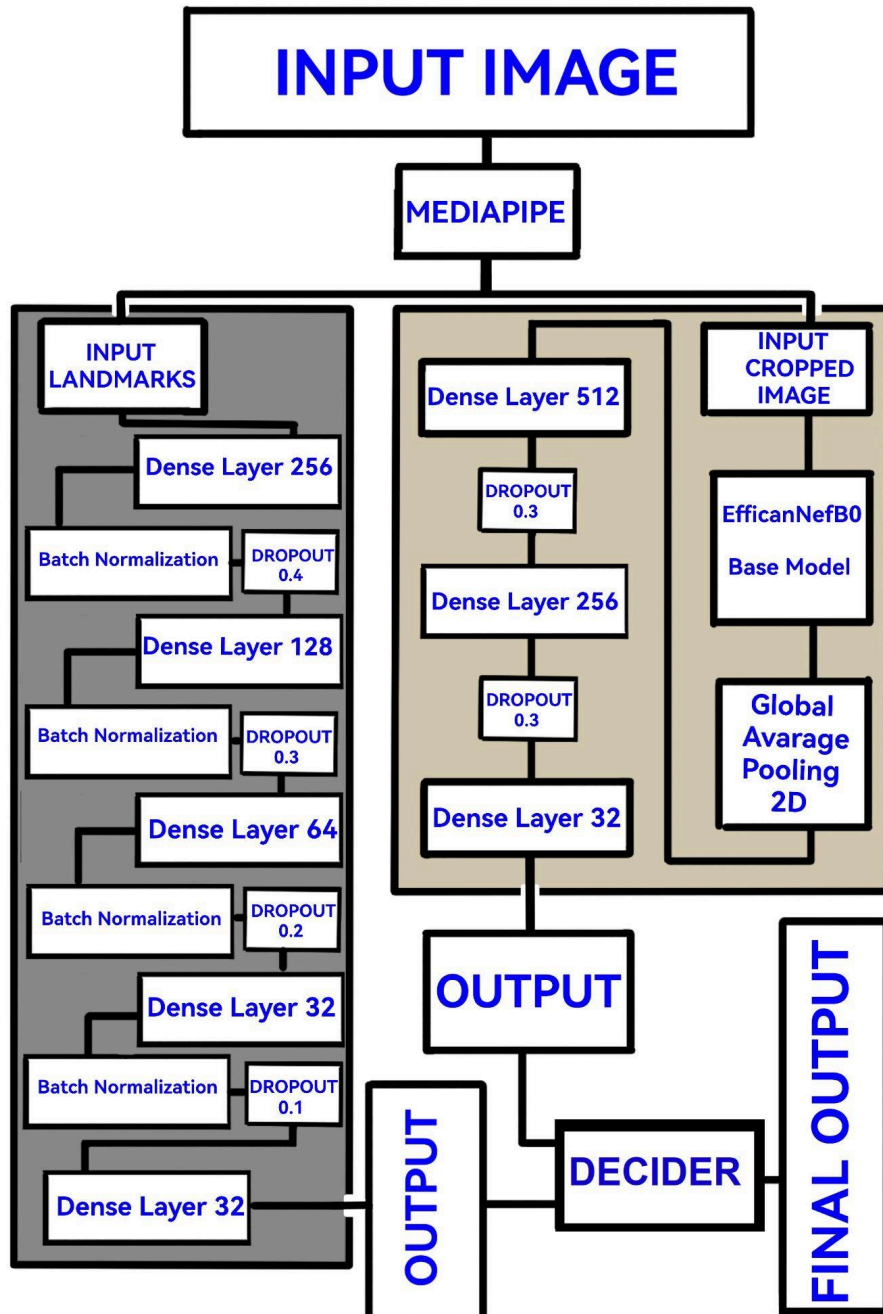
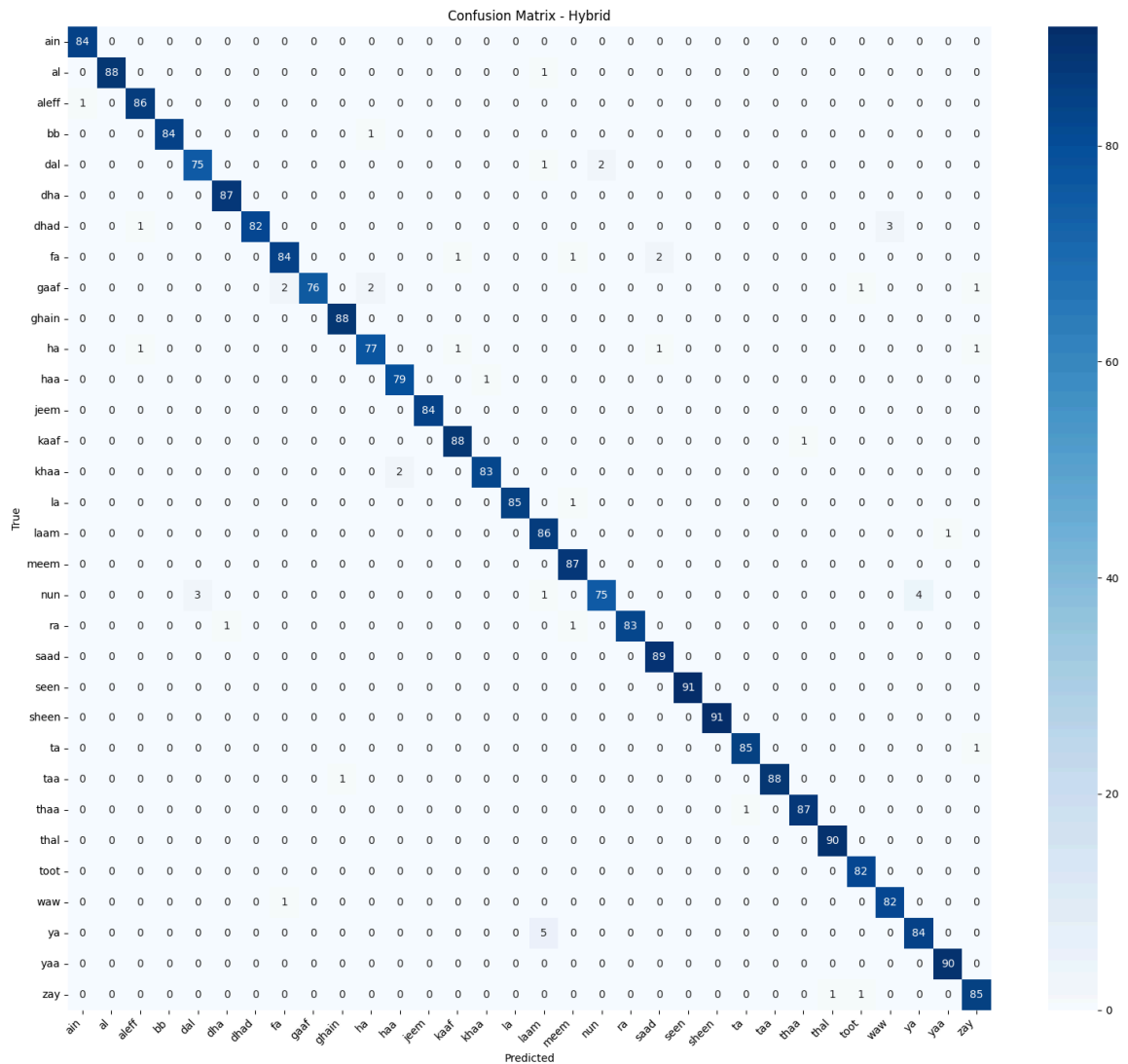


Figure 5.7 - Hybrid Model Evaluation & Confusion

Accuracy	Precision	Recall	F1 Score
0.9815	0.9818	0.9815	0.9814



CHAPTER 6: CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this project, we have successfully reached a robust model that can detect sign language in various conditions, which will be used in the future with a mobile application to help bridge the gap between deaf and hearing communities. Both EfficientNetB0 and Dense Neural Network models have achieved great results individually. with the EfficientNetB0 performing slightly better by about 1%, while the Dense Neural Network with the Landmarks finished training and achieved close results with very less time. Combining both have boosted the results even further, achieving a 98% accuracy in detecting Arabic sign language.

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>