



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	8
1.4 Scope and Limitations.....	8
CHAPTER 2: LITERATURE REVIEW	9
2.1 Introduction	9
2.2 History of Sign Language Recognition	9
2.3 Related Work	9
CHAPTER 3: DATASET	
3.1 Dataset Description	11
3.2 Preprocessing	12
CHAPTER 4: PREPOSED APPROACH	14
4.1 Framework	14
4.2 Models architecture	14
4.3 Data Preprocessing And Augmentation	15

4.4 Hyperparameter Tuning	15
CHAPTER 5: EXPERIMENTAL RESULTS AND DISCUSSION	16
5.1 Resnet50 Model	16
5.2 EfficientNetB0 Model	18
5.3 Dense Neural Network Model	22
5.4 The Hybrid Model	25
5.5 Models Results And Evaluation	28
CHAPTER 6: CONCLUSION AND FUTURE WORK	30
6.1 Conclusion	30
6.2 Future Work	30

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
- **HMM:** Hidden Markov Model

List of Figures

3.1 Dataset Samples	12
5.1 Fa & Gaaf Samples	16
5.2 Resnet50 Evaluation & Confusion Matrix	17
5.3 Similar Sign Gestures Examples	19
5.4 EfficantNetB0 Evaluation & Confusion Matrix	21
5.5 Dense Neural Network Evaluation & Confusion Matrix	23
5.6 Dense Neural Network Training And Validation Accuracy Plot	24
5.7 Hybrid Model Architecture	26
5.8 Hybrid Model Evaluation & Confusion Matrix	27
5.9 Models Results	28
5.9 Models Comparision	29

CHAPTER 1: INTRODUCTION

1.1 Background and motivation

The deaf community faces significant communication barriers in their daily interactions with hearing individuals. While Arabic sign language (ArSL) is used as the primary tool for communication for the deaf community, daily communication remains a challenge for them because ArSL is not widely known by hearing people.

Sign language recognition is one of the domains that have been transformed by artificial intelligence. A lot of studies have shown the potential of using AI for sign recognition as a solution to bridge the gap between deaf and hearing people [12].

This motivated us to develop an AI powered system for the Arabic Alphabet Sign Language .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 that can perform robustly in real word settings, while being accessible, user-friendly, and practical for everyday use.

1.3 Research Objectives

The research aims to propose, develop, and evaluate 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, utilizing deep learning and computer vision methodologies, while maintaining real-time processing capabilities.

1.4 Scope and Limitations

The current approach focuses on detecting and translating a static hand sign to an Arabic alphabet character by training and evaluating multiple models to find the optimal model that would perform robustly in real word settings (different backgrounds, gender, hand sizes and colors). This includes critically analysing the dataset to find its limitations and come up with strategies to improve it for our task.

One of the limitations is that these models were trained with static images gestures due to the lack of Saudi Sign Language dataset for words and sentences. We were able to get a very small dataset with dynamic gestures, but the size of this data was not sufficient enough to create a robust model.

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 History of Sign Language Recognition

Sign recognition technologies have been rapidly evolving since the 1980. In the early studies, the devices used for hand movement recognition were special gloves and sensors [13]. Then in 1996, Starner et al. [14] used Hidden Markov Modeling (HMM) with a colored camera to track the hand and interpret its gesture. As mentioned by Starner et al In early systems, HMM was a popular choice because of its ability to recognise dynamic sign language gestures. However, their ability to recognise static gestures were limited due to the absence of motion in static images. Around the 2000-2010, when cameras became affordable, the focus shifted to vision based recognition for hand detection [15]. The advancement of deep learning around 2014 has caused a huge shift in this domain. Convolutional Neural Networks (CNNs) were introduced and used for feature extraction and have outperformed previous approaches, achieving higher accuracy on alphabet and digital signs [16].

2.3 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 ”لنا” 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.

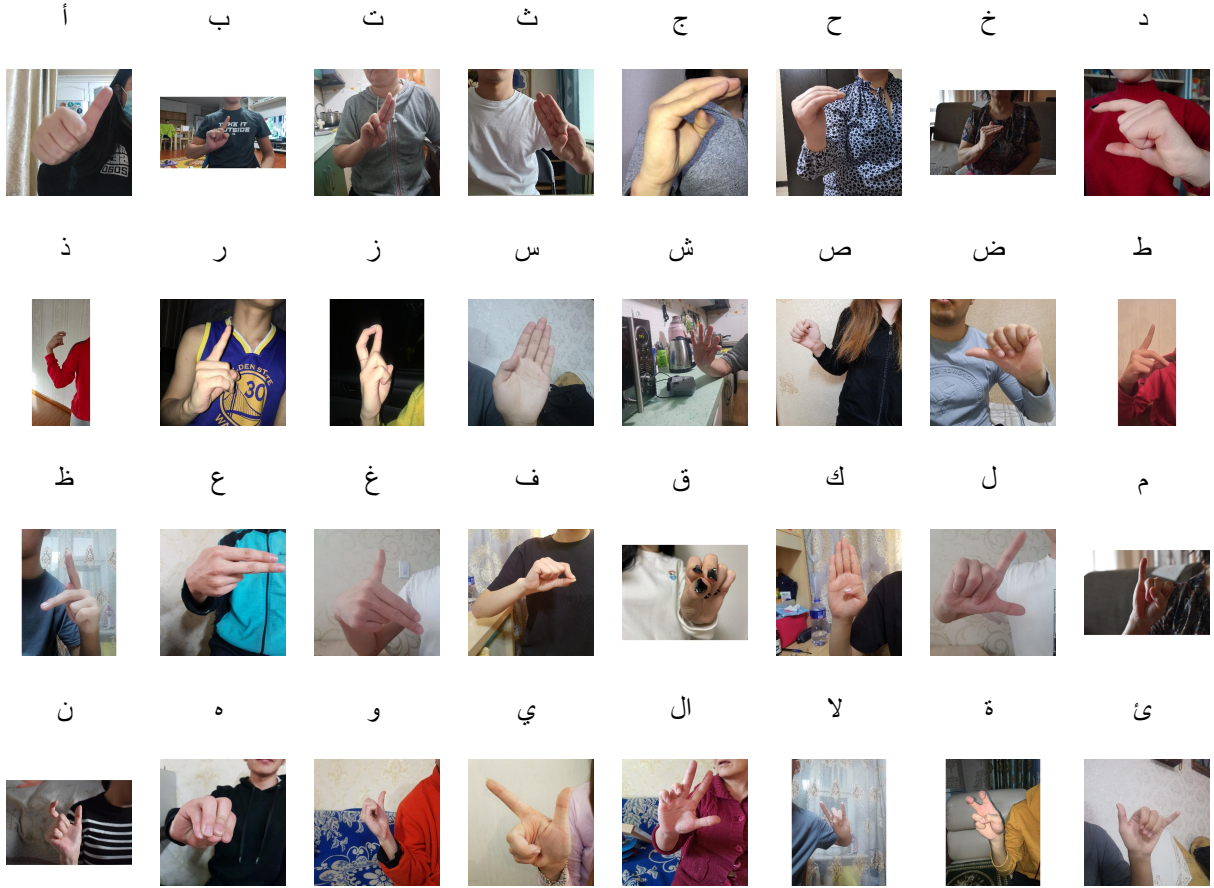
To address dynamic gestures, hybrid architectures combining CNNs with sequential models have emerged. Noor et al. (2024) [3] proposed a CNN-LSTM framework for real-time ArSL recognition. They have achieved 94.40% accuracy for static gestures, and 82.70% for dynamic ones. They have trained the models 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.

Other studies have tried to use pre-trained models like resNet, InceptionV3, MobileNetV2 and EfficientNetB7, and were able to achieve 90%+ accuracy [17][18][19][20].

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.

Figure 3.1 - Dataset Samples.



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

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 5.2. Looking at the confusion matrix, the most mispredicted classes were Fa and Gaaf. This model had some difficulty with these due to their sign similarity as shown in figure 5.1.

Figure 5.1 - Fa & Gaaf Samples

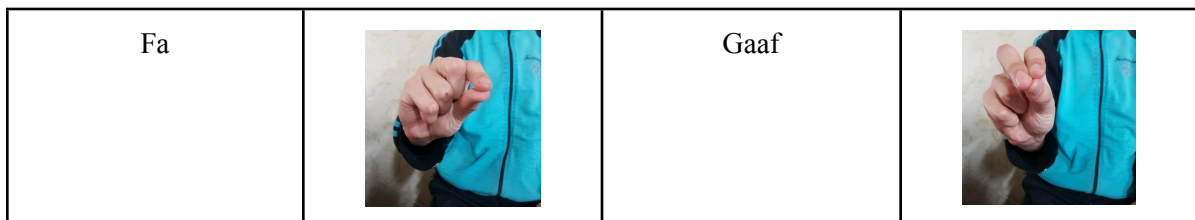
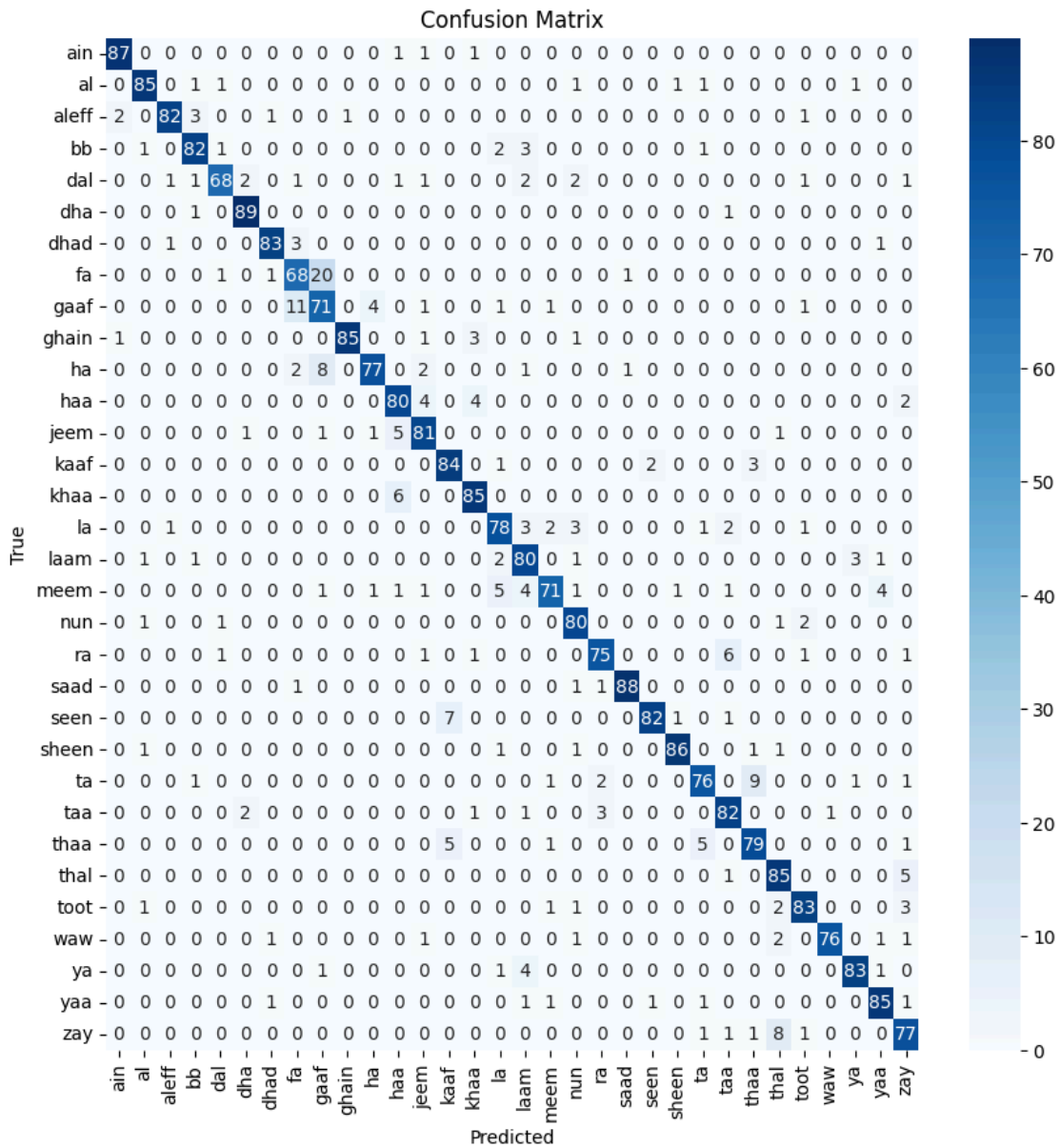


Figure 5.2 - Resnet50 Evaluation & Confusion Matrix

Accuracy	Precision	Recall	F1 Score
0.8974	0.8999	0.8974	0.8978

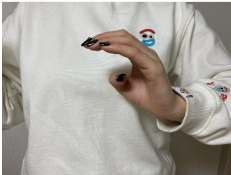


5.2 EfficientNetB0 Model

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 5.3 for some examples the model struggled with.

Figure 5.3 - 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

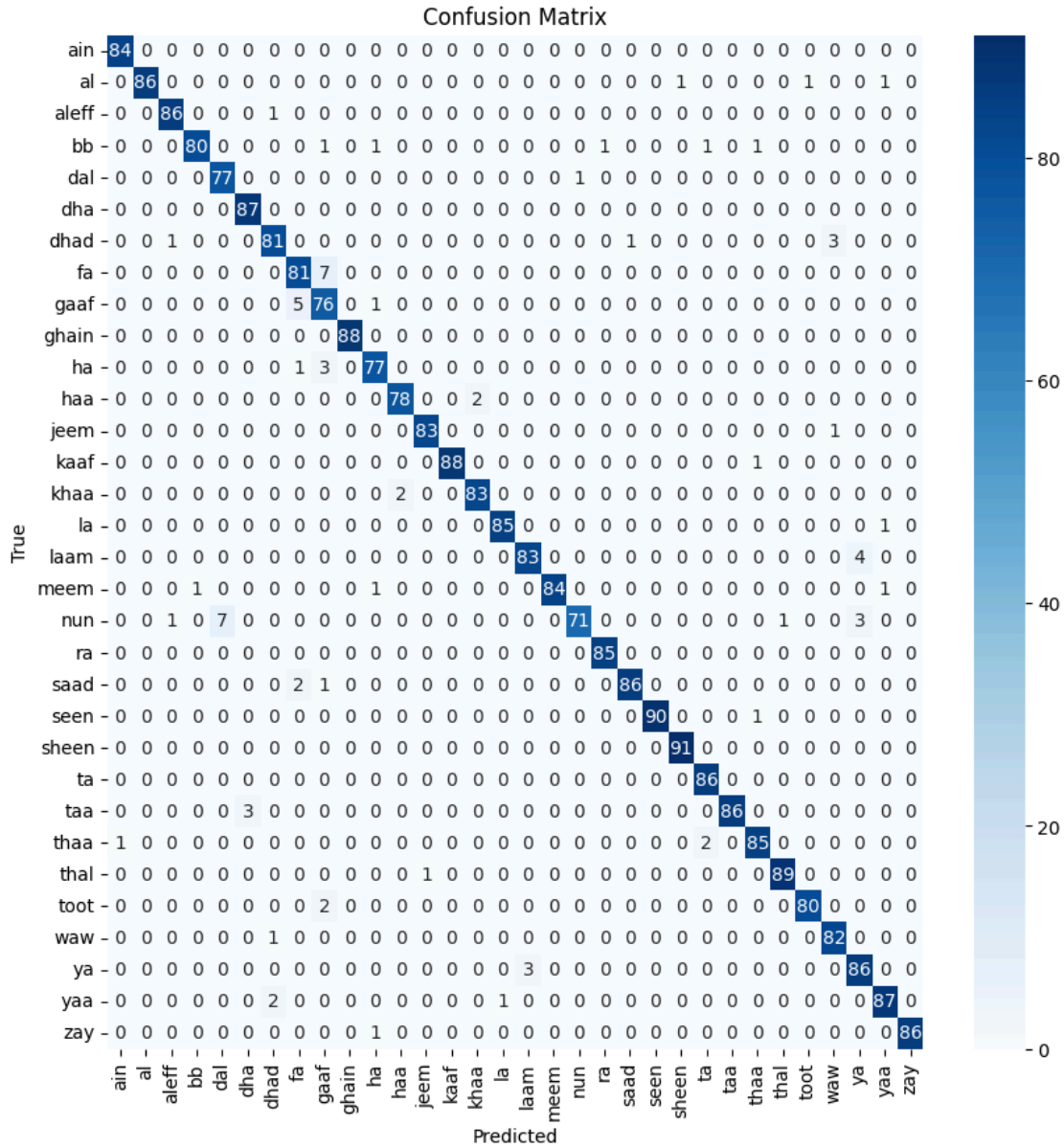
google to first detect the hand, then crop the image around its coordinates to make the image more focused on the hand and ignore the rest.

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 preproccing by actually looking though 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 5.4, the classes that had the most mis-predictions are Nun being mispredicted as Dal because of the similarity of the sign itself as shown previously in Figure 5.3. Also, by comparing the results to Resnet50 results, we can conclude that the model does have a better handle on both Fa and Gaaf compared to the Resnet50 model.

Figure 5.4 - 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. From the confusion matrix shown in **figure 5.5**, we can see that this model had a better grasp of classifying Fa and Gaaf signed, achieving 0 wrong prediction to the other class compared to the previous results. Although this is much better than the previous model, it performed less in other cases, like more Nun mispredicted as Dal, and Fa mispredicted as Saad.

Figure 5.5 - Dense Neural Network Evaluation & Confusion Matrix

Accuracy	Precision	Recall	F1 Score
0.9633	0.9641	0.9633	0.9633

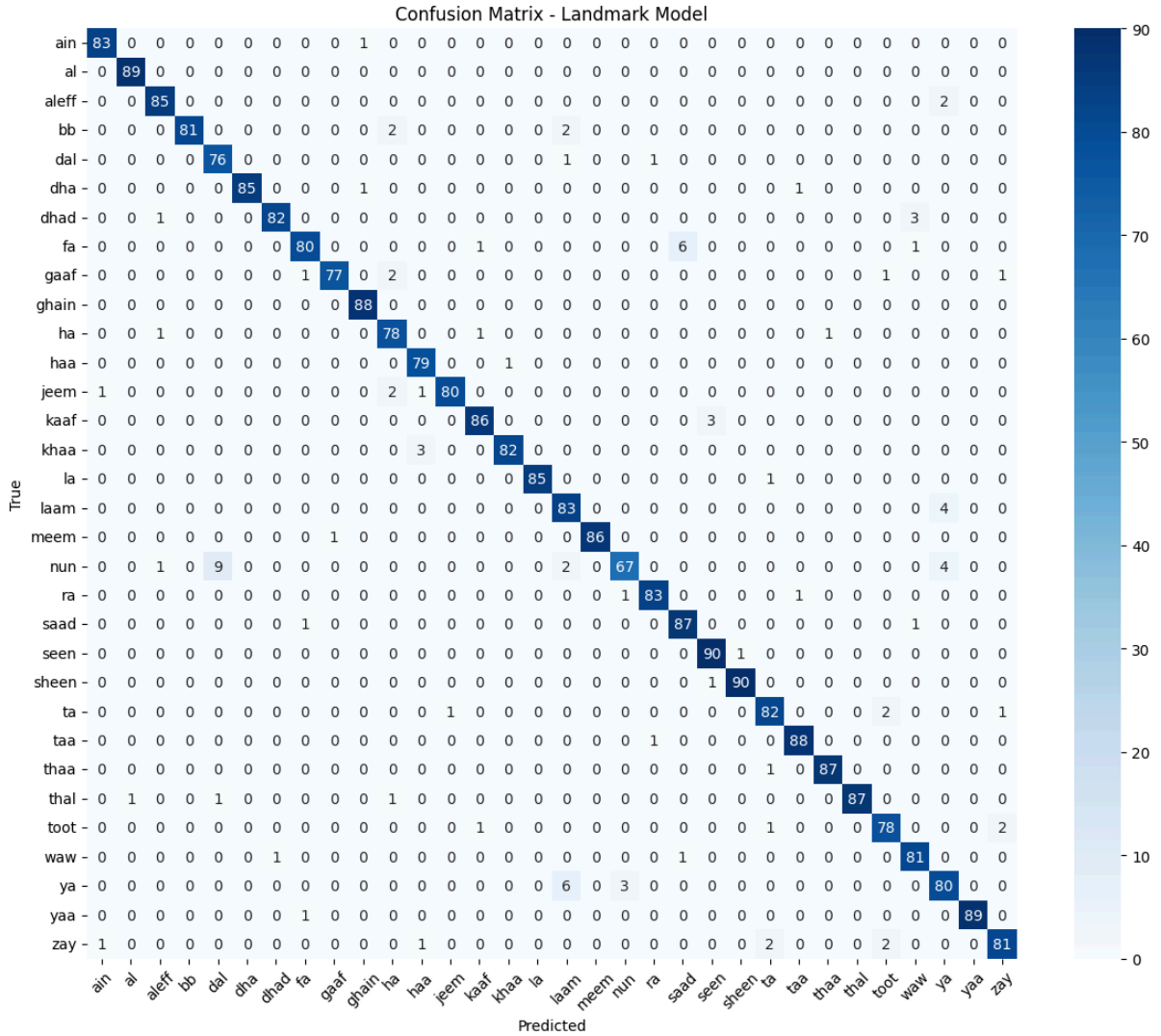
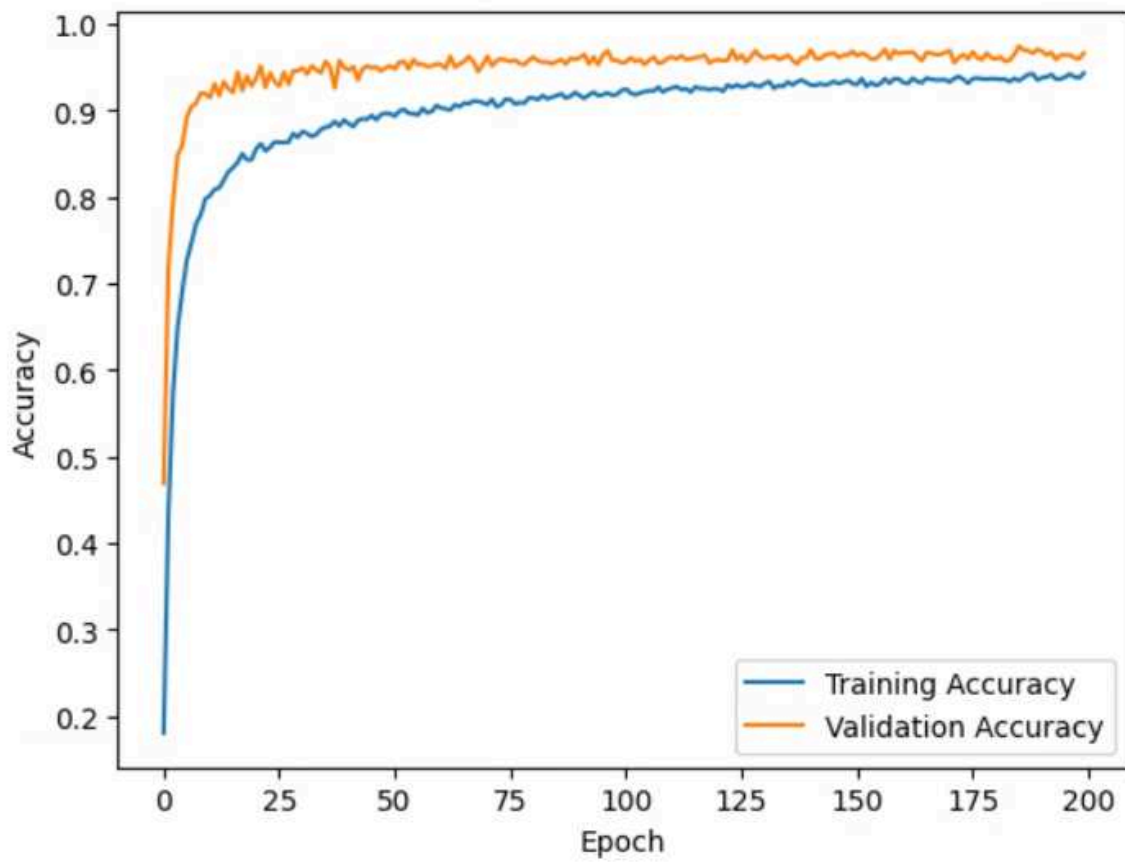


Figure 5.6 - Dense Neural Network Training And Validation Accuracy accuracy



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 some difficulty distinguishing between “taa” and “dha”, but had a better grasp of “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 Hybrid model was made and the architecture of the model is shown in Figure 5.7. It takes the image and passes it on to MediaPipe. This will give us two outputs. The first one is the cropped image, and the second one is the landmark text file. In the next step, the image gets passed to the EfficientNetB0 model, while the text landmark gets passed to the Dense Neural Network Model simultaneously. After that, both outputs get passed to the desider method that will determine the final output.

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 as shown in figure 5.8. It had achieved better results overall compared to each model individually.

Figure 5.7 - Hybrid Model Architecture

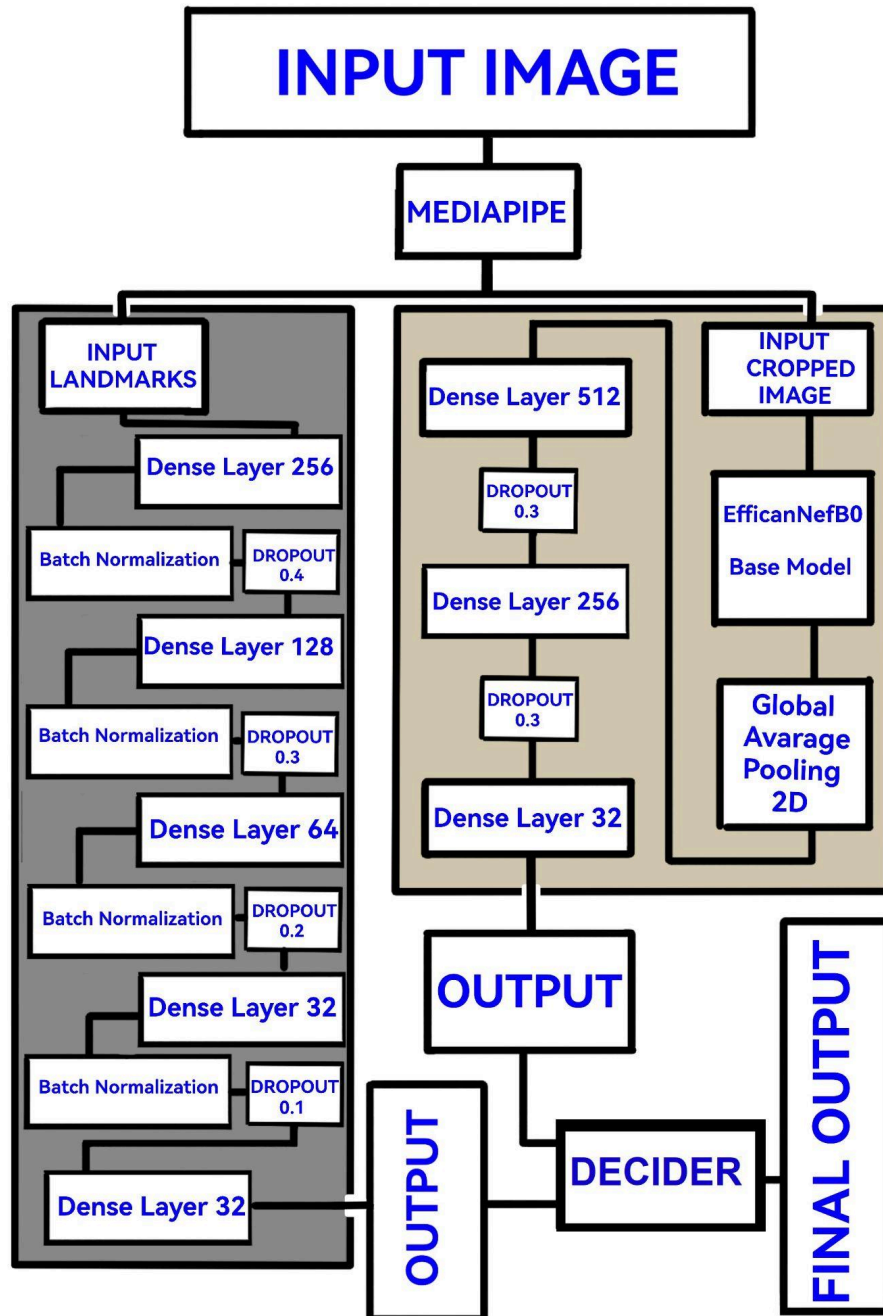
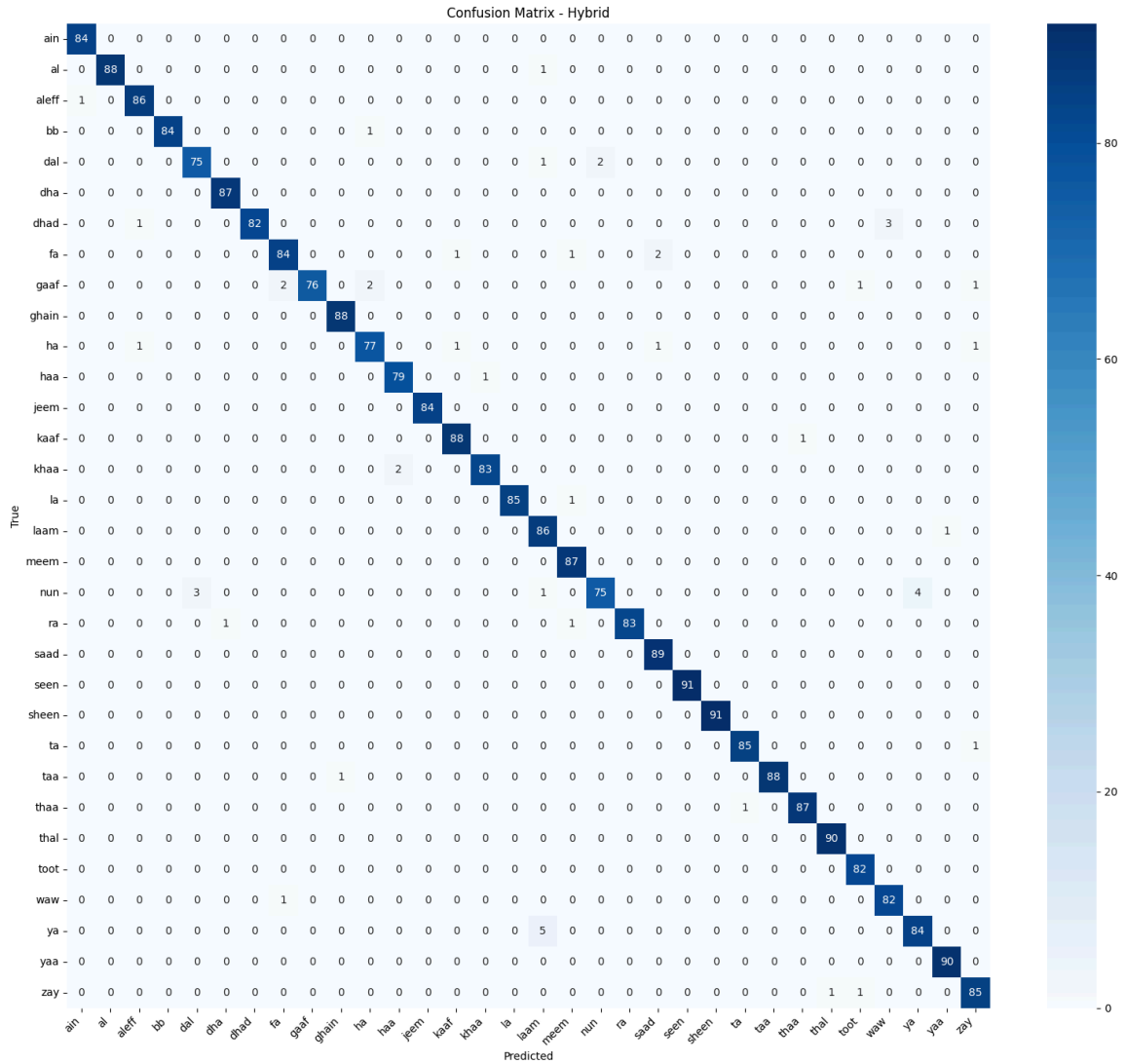


Figure 5.8 - Hybrid Model Evaluation & Confusion

Accuracy	Precision	Recall	F1 Score
0.9815	0.9818	0.9815	0.9814



5.5 Models Results And Evaluation

In this research, we have trained and tested 4 different models to classify images for arabic alphabet sign language. The models were evaluated using accuracy, precision, recall, and F1 score as shown in the figure 5.9. Resnet50 yielded moderate performance, achieving an accuracy of 89%. while EfficianNetB0 achieved better results with the same setup as Resnet50 with less epochs, and achieved an even higher accuracy with more preprocessing of the dataset, reaching an accuracy of 97%. The Dense Neural Network had the best training time and used the least resources out of the previous models, while achieving a substantially high performance of 96% accuracy, that is comparable to the Efficto the EfficiantNetB0 Model, indicating that both approaches (image feature extraction and hand landmark) are very effective for this task. The hybrid model combined both approaches (EfficiantNetB0 and Dence Neural Network) to take the best of both and achieve a solid 1% increase in accuracy for a total of 98% accuracy.

Figure 5.9 - Models Results

Model	Accuracy	Precision	Recall	F1 Score
Resnet50	0.8974	0.8999	0.8974	0.8978
EfficiantNetB0	0.9713	0.9721	0.9713	0.9714
Dence Neural Network	0.9633	0.9641	0.9633	0.9633
The Hybrid	0.9815	0.9818	0.9815	0.9814

In figure 5.10 below, we compare our model to other similar ones from previous studies.

Figure 5.10 - Models Comparision

Model Reference	Dataset Description	Dataset Size (images)	Validation Accuracy
[7]	25 words	200	85%
[9]	Alphabets and Number	900	100%
[10]	Alphabets	1147	84.68%
[11]	Alphabets	900	70%-80%
[1]	Alphabets, Number and One Word	27,301	99.47%
Our Hybrid Model	Arabic alphabet + 4 more signs	14,202	98.15%

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 first step would be to start to collect a dataset for the dynamic gesture that makes up a word. Then refine and clean the dataset to be usable for deep learning. Then 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>
7. M. ElBadawy, A. S. Elons, H. A. Shedeed, and M. F. Tolba, "Arabic sign language recognition with 3d convolutional neural networks," in *2017 Eighth International Conference on Intelligent Computing and Information Systems (ICICIS)*, Dec 2017, pp. 66–71.
8. S.Hayani,M.Benaddy,O.ElMeslouhi, and M.Kardouchi, "Arab sign language recognition with convolutional neural networks," in *2019 International Conference Of ComputerScience and Renewable Energies (ICCSR)*, July 2019, pp. 1–4.
9. M. Taskiran, M. Killioglu, and N. Kahraman, "A real-time system for recognition of american sign language by using deep learning," in *the 2018 41st International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2018, pp. 1–5.
10. M.Hossen, A.Bhuiyan, A.Govindaiah, S.Sultana, and "Bengalisignlanguage recognition using deep convolutional neural network," in *2018 Joint 7th International Conference on Informatics, Electronics & Vision (ICIEV) and 2018 2nd International Conference on Imaging, Vision & Pattern Recognition (icIVPR)*. IEEE, 2018, pp. 369– 373.
11. I. Makarov, N. Veldyaykin, M. Chertkov, and A. Pokoev, "Russian recognition," in *sign language dactyl 2019 42nd International Conference on Telecommunications and Signal Processing (TSP)*. IEEE, 2019, pp. 726–729.
12. Alayed, A. (2024). Machine Learning and Deep Learning Approaches for Arabic Sign Language Recognition: A Decade Systematic Literature Review. **Sensors**, 24(23), 7798.
<https://doi.org/10.3390/s24237798>

13. Koller, O. (2020). *Quantitative survey of the state of the art in sign language recognition* (arXiv:2008.09918). arXiv. <https://arxiv.org/abs/2008.09918>
14. Starner, T., Weaver, J., & Pentland, A. (1998). *Real-time American sign language recognition using desk and wearable computer based video. IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12),
https://sites.cc.gatech.edu/home/thad/p/journal/real-time-asl-recognition.pdf?utm_source=chatgpt.com
15. Tolentino, L. K. S., Serfa Juan, R. O., Thio-ac, A. C., Pamahoy, M. A. B., Forteza, J. R. R., & Garcia, X. J. O. (2019). *Static sign language recognition using deep learning. International Journal of Machine Learning and Computing*,
https://www.researchgate.net/publication/337285019_Static_Sign_Language_Recognition_Using_Deep_Learning
16. Gangal, A., Kuppahally, A., & Ravindran, M. (2024). *Sign language recognition with convolutional neural networks. Stanford CS231n Final Project Report*.
<https://cs231n.stanford.edu/2024/papers/sign-language-recognition-with-convolutional-neural-networks.pdf>
17. Lahiani, H., & Frikha, M. (2024). *Exploring CNN-based transfer learning approaches for Arabic alphabets sign language recognition using the ArSL2018 dataset. International Journal of Intelligent Engineering Informatics*.
<https://doi.org/10.1504/IJIEI.2024.138858>
18. Balat, M., Awaad, R., Adel, H., Zaky, A. B., & Aly, S. A. (2024, October 1). *Advanced Arabic*

alphabet sign language recognition using transfer learning and transformer models
[Preprint]. arXiv. <https://arxiv.org/abs/2410.00681>

19. Saleh, Y., & Issa, G. F. (2020). *Arabic Sign Language Recognition through Deep Neural Networks Fine-Tuning*. *International Journal of Online and Biomedical Engineering*, 16(5), 71–83. <https://doi.org/10.3991/ijoe.v16i05.13087>
20. Alharthi, N. M., & Alzahrani, S. M. (2023). *Vision Transformers and Transfer Learning Approaches for Arabic Sign Language Recognition*. *Applied Sciences*, 13(21), Article 11625. <https://doi.org/10.3390/app132111625>