

UNIVERSITY OF BAHRAIN
College of Information Technology



Real Time Translation of Arabic Sign Language to Arabic Text Using Deep Learning Model

A Thesis Submitted in Partial Fulfilment of the Requirements for the
M.Sc. Degree in Machine Learning and Computational Intelligence.

Submitted by
Shaikhah Ibrahim Almana
ID Number: 202101772

Supervised by
Dr. Nabil Hewahi
Professor
University of Bahrain

Co-Supervised by
Dr. Alauddin Alomary
Associate Professor
University of Bahrain

Kingdom of Bahrain
April 2023

Defense Committee's Approval

University of Bahrain
Deanship of Graduate Studies
And scientific Research



جامعة البحرين
عمادة الدراسات العليا والبحث العلمي

Thesis Modifications Approval Form

Student Name: Shaikhah Almana	Student ID: 202101772
Department: Computer Science	College: Information Technology
Program: Master of Machine Learning and Computational Intelligence	
Thesis Title: Real Time Translation of Arabic Sign Language to Arabic Text Using Deep Learning Model	
Supervisor Name: Nabil Hewahi, Aladdin Alomary	Student GPA: 3.72

We would like to report that the student has carried out all the required modifications made by the discussion committee. Upon this, we recommend awarding the student a grade ^{PASS} Choose an item. in the thesis course number ITML607 based on the Discussion Committee recommendation. ^{PASS} Choose an item.

Endorsement:

	Name	Academic Title	Signature	Date
1.	Dr. Nabil Hewahi, Dr. Aladdin Alomary	Advisor	Click here to enter text. <i>Nabil Hewahi Alomary</i>	24/6/2023
2.	Dr. Faisal Alkhateeb	Internal Examiner	<i>Faisal Alkhateeb</i>	
3.	Dr. Nabil Hewahi	Department Coordinator	<i>Nabil Hewahi</i>	
4.	Dr. Ahmad Fahad	Department Chairman	Click here to enter text.	19/9/2023
5.	Dr. Hesa Aljunaid	College Dean	Click here to enter text. <i>Hesa Aljunaid</i>	

7 Approved at the University Council 6th meeting, 2013
In accordance with Decision No. 736/2013 dated 14/3/2013
Modified by University Council Decision No. 2043/2013 dated 20/11/2013.

Acknowledgement

I would like to sincerely thank my supervisor Prof. Nabil Hewahi and co-supervisor Assoc. Prof. Alauddin Alomary their invaluable help and direction throughout my master's program. They provided me with the information and assurance I needed to do this research and produce this thesis.

I also want to express my gratitude to my family and friends for their love and support throughout this process. This experience would not have been possible without them.

Dedication

This work is dedicated to my family, especially my father. For supporting me and helping me gaining a high-quality education. Also, my close friends. They have always supported me throughout my study.

Abstract

Hearing-impaired people communicate with each other through sign language (SL). Deaf and hard-of-hearing people use sign language (SL) to communicate their emotions and ideas. There are numerous sign languages, and each sign language has a variety of regionally based dialects. The lack of communication between hearing people and people with hearing impairments is a problem affecting the entire globe. This thesis proposes an application that recognizes Arabic Sign Language (ArSL) using the convolutional neural network (CNN) model in real time. ArSL2018, which is made up of 32 categories and more than 5,000 images, is used to train the CNN model. The performance of the model achieved a weighted-average, F1-score of 97%, the macro-average F1-score was 97%, and accuracy reached 97%. Next, real-time recognition was implemented using OpenCV. It started by capturing a real time frame. After that, the pretrained CNN model is fed the capture image to predict the letter. The application was implemented successfully, and real-time recognition was achieved.

Table of Contents

<i>Defense Committee’s Approval</i>	<i>a</i>
<i>Acknowledgement</i>	<i>b</i>
<i>Dedication</i>	<i>c</i>
<i>Abstract</i>	<i>d</i>
<i>Table of Contents</i>	<i>e</i>
<i>List of Figures</i>	<i>g</i>
<i>List of Tables</i>	<i>h</i>
<i>Publications</i>	<i>i</i>
<i>List of Abbreviation</i>	<i>j</i>
1 Introduction	2
1.1 Sign language.....	2
1.2 Problem Statement	2
1.3 Research Questions.....	3
1.4 Objectives	3
1.5 Research Significance	3
1.6 Scope and Limitations	4
1.7 Thesis Organization.....	4
2 Background and Literature review	6
2.1 Background	6
2.1.1 Image Processing Approaches	6
2.1.2 CNN Architecture	10
2.2 Literature review	16
2.2.1 Non-Arabic Sign Language	16

2.2.2	Arabic Sign Language	18
3	<i>Methodology</i>.....	23
3.1	The Proposed Architecture.....	23
3.1.1	CNN Proposed Architecture	23
3.2	Real Time implementation	26
4	<i>Experimentations and Results</i>.....	29
4.1	Dataset	29
4.1.1	Dataset description.....	29
4.1.2	Data Preprocessing	30
4.2	Evaluation Metrics	31
4.3	Experimentations	33
4.3.1	Development Requirements.....	33
4.4	Results	34
4.4.1	Application performance	34
4.4.2	Real Time Application.....	38
4.4.3	Discussion.....	38
5	<i>Conclusion and Recommendation</i>.....	42
5.1	Conclusion	42
5.2	Recommendation	42
	<i>References</i>.....	43
	<i>Appendix</i>.....	51
	Regularization.....	52
	Backpropagation.....	54
	الخلاصة.....	55

List of Figures

Fig.2.1 Representation of the effect of binarization.....	7
Fig.2.2 Representation of the effect of shearing augmentation	8
Fig.2.3 Representation of the effect of zooming augmentation	8
Fig.2.4 Representation of the effect of horizontal flip	9
Fig.2.5 Mathematical representation of a neuron (Liu, 2017).	10
Fig.2.6 3-layer artificial neural network with two hidden layers (Liu, 2017).....	11
Fig.2.7 Representation of basic CNN architecture (Balaji, 2020).	13
Fig.2.8 Representation of general convolutional operation (Li , Zhou , Wang, Shen, & Ran, 2021).	14
Fig.2.9 Maxpooling example.	15
Fig.2.10 NN with and without dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).....	15
Fig.3.1 Architecture for the proposed approach.....	23
Fig.3.2 Illustration of proposed CNN model.	24
Fig.3.3 Real time workflow	27
Figure 4.1 illustration of evaluation measurements for the training	35
Figure 4.2 illustration of evaluation measurements for the training	36
Fig.4.3 A try out of the final application result.	38
Figure 4.4 (a) Ha sign (b) Gaaf sign	39
Fig.4.5 Compression of the three-model supporting real time from Table 4.2.....	40
Fig.0.1 Algorithm of computing Adam.....	53

List of Tables

Table 3.1 Data, Gesture and Number of Images	29
Table 3.2 The configuration and the number of parameters in the proposed architecture	26
Table 3.3 Confusion Matrix	31
Table 4.1 Output of the classification report for the testing	37
Table 4.2 Models Comparison	40

Publications

- I. Almana, S., & Al-Omary, A. (2022). Real-time Arabic Sign Language Recognition using CNN and OpenCV. *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 32--36.

List of Abbreviation

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Networks
ArSL	Arabic Sign Language
ASL	American Sign Language
CNN	Convolution Neural Network
DL	Deep Learning
FE	Feature Extraction
FN	False Negative
FP	False Positive
GPUS	Graphics Processing Units
ISL	Indian Sign Language
ML	Machine Learning
RNN	Recurrent Neural Networks
ROI	Region Of Interest
SL	Sign Language
TN	True Negative
TP	True Positive

Introduction

1 Introduction

In this chapter six subsection is stated. First, a general background discussing sign language, machine learning, deep learning and convolution neural network is presented. Second, the problem statement is specified. Third, the objective of the thesis is listed. Fourth, the research significance is stated. Fifth, scope and limitations are listed. Finally, thesis organization is presented.

1.1 Sign language

Individuals who have hearing loss use sign language (SL) as their primary form of communication. Hearing-impaired people use sign language to communicate their emotions and ideas. American Sign Language (ASL; Liddell, 2003) and Arabic Sign Language (ArSL; Abdel-Fattah, 2005) are two of many types of SLs, which are similar to spoken language. Every SL also has several distinct dialects. For instance, ArSL has dialects from the Gulf, Palestine, Egypt, Tunisia, etc. Even though ArSL changes from country to country, the numbers and alphabet in ArSL are static (Ismail et al., 2021). Nonmanual signs and manual signs are the two main parts of SL (Mukushev et al., 2020). Hand location, hand shape, and hand orientation are used to understand manual signals. Nonmanual signs are interpreted using body language and facial reactions. The essential information is delivered using manual signs, whereas nonmanual signs are used to help stress and clarify the manual signs. This study is focused on ArSL manual signs.

1.2 Problem Statement

The dearth of interpreters for ArSL makes it essential to have real-time systems that interpret ArSL to facilitate daily communication with deaf people. In Arab countries there are over 11 million deaf individuals (Reading

the Signs: Diverse Arabic Sign Languages, 2014). In Saudi Arabia alone, there are over 400,000 people with hearing impairments and only 103 licensed interpreters (Saudi Sign Language Association, 2022)—a ratio of one interpreter for every 3,883 hearing-impaired people. Although sign language differs in practice from one Arabic country to another, the alphabets are all the same. This work aimed to create a real-time translation application that translates ArSL letters into Arabic text using a deep learning model with a better performance than existing models.

1.3 Research Questions

- How to improve real-time recognition for ArSL?
- Can an application be implemented to perform real-time recognition?
- Can the application enhance communication between hearing and deaf people?

1.4 Objectives

- Implement a real-time ArSL recognition application for the Arabic alphabet.
- Enhance communication between hearing and deaf people.
- Improve the performance of currently available models.

1.5 Research Significance

Real-time recognition for ArSL is not explored enough. To the knowledge of the researcher, there are only three papers considering the real-time recognition for ArSL (Kamruzzaman, 2020; Alsaadi et al., 2022; Almana & Al-Omary, 2022). Also, this study will help the Arab community by improving communication between hearing and deaf individuals.

1.6 Scope and Limitations

- This model analyzes static hand gesture using CNN in real time.
- This model recognize sign and convert it to text only.
- This work is applied on the ArSL2018 dataset (Latif, Mohammad, Alghazo, AlKhalaf, & AlKhalaf, 2019). It contains images of ArSL letters.
- Accuracy, a weighted average F1-score, and a macro-averaged F1-score are used to evaluate this application.

1.7 Thesis Organization

This thesis is structured as follows: In Chapter 2, we provide a detailed background describes image processing approaches and CNN architecture are exhibited. Next, a general literature review on previous work for non-Arabic and ArSLs in ML, CNN and real time translating is presented. Chapter 3 contains our design, implementation, and experiments of the proposed model that translates Arabic Sign Language to Arabic Text. It contained three parts: The first part presented a detailed description of the dataset. The second part declared the proposed model design and illustration. In the final part, described the evaluation metrics. Chapter 4 provides Results, Discussion, and recommendations of our work.

Background and Literature review

2 Background and Literature review

In this chapter a detailed background explains image processing approaches and CNN architecture are exhibited. After that, a literature review on previous work for non-Arabic and ArSLs in ML, CNN and real time translating is presented.

2.1 Background

Within this section, we state a brief explanation and give a simple background on some important parts used in our proposed approach. Since we are dealing with images, we present here some of the image processing techniques needed in our model. Also, give an explanation of CNN as it is the ML model which was adopted in our proposed approach.

2.1.1 Image Processing Approaches

Preprocessing data is a process applied to manipulate raw data to enhance data and prepare it for ML (Zelaya, 2019). Data preprocessing is an essential step in ML as it improves the model's performance. Numerous techniques are used to perform data preprocessing depending on the data type. This work is based on images, so image preprocessing and image augmentation are used in it. Also, another important data preprocessing step is data splitting.

2.1.1.1 Image preprocessing

Image preprocessing is an essential step that uses numerous operations to format images (Krig & Krig, 2016). Image preprocessing is used to prepare the data before it is used in the model. If the image is not preprocessed probably, it may cause the model to not perform as expected. There are multiple image preprocessing techniques such as brightness corrections,

image filtering, image segmentation, image binarization, etc. This work uses only one image preprocessing technique which is binarization.

2.1.1.1.1 Binarization

It is the process of converting a greyscale image into a black-and-white image (Chaki, et al., 2014). It is used as a technique operated to cut down the information from 256 shades into two shades of black and white. This process is done using a threshold. A threshold is an image preprocessing technique that works by setting a specified value x . Then, the specified value x is compared to every pixel in an image. The value of the pixel is changed to 1 when it exceeds or is equal to the value x . On the other hand, a pixel's value is altered to 0 when it is lower than the value x . The binarization representation is shown in Fig.2.1.

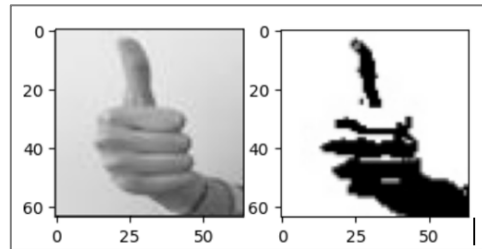


Fig.2.1 Representation of the effect of binarization

2.1.1.2 Image augmentation

Image augmentation is manipulating the pixels in the image to create multiple versions of an image (Shorten & Khoshgoftaar, 2019). Image augmentation is used to expose the model to a variety of images for training. Some of the image augmentation techniques include but are not limited to rotation, brightness, rescaling, zooming, contrast, edge enhancement, etc.

In this work, some image augmentation algorithms are used to preprocess the dataset, including rescaling, shearing, zooming, and horizontal flip.

2.1.1.2.1 Rescaling

The process of downscaling pixel values from the 0 to 255 range to the 0 to 1 range is known as rescaling or normalizing. Rescaling is used to separate the data well in the space to ease the learning process for the algorithm.

2.1.1.2.2 Shearing

Shearing in image augmentation is the process of destroying the image along the x or y-axis. Shearing is used to create angled perceptions of an object. It is used in images to mimic how humans view things from multiple angles. Fig.2.2 visualize the effect of shearing.

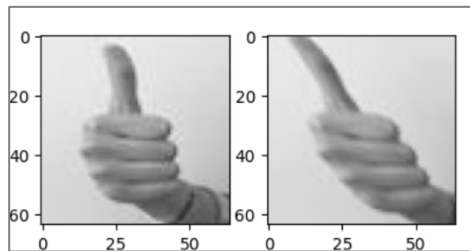


Fig.2.2 Representation of the effect of shearing augmentation

2.1.1.2.3 Zooming

Zooming is one type of image augmentation that randomly zooms an image. It is used to create a new image with multiple zooming levels. Zooming is used in images to represent how an object can be close or far from a camera. A visualization of the zooming technique is shown in Fig.2.3.

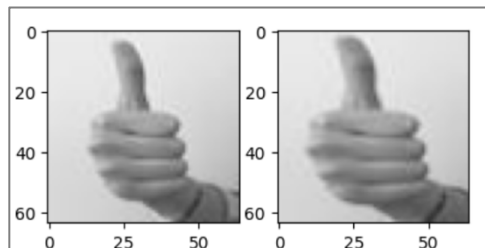


Fig.2.3 Representation of the effect of zooming augmentation

2.1.1.2.4 Horizontal Flip

Horizontal flip augmentation is the procedure of reversing and rotating the image on the horizontal axis. It is done by reversing the pixels' location. Horizontal flip is used to mimic how an object can be seen in normal shape or flipped by a mirror or in this work for example the sign from the left and right hand. The horizontal flip representation is shown in Fig.2.4.

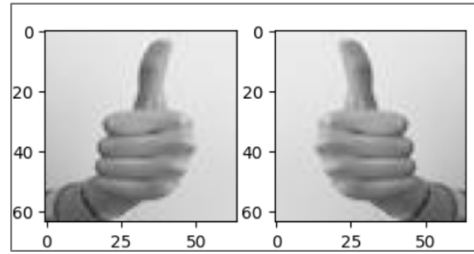


Fig.2.4 Representation of the effect of horizontal flip

2.1.1.3 Data splitting

Data splitting is an important data preprocessing step. It is a method where the data is divided into multiple subsets. In preparation for ML dataset is commonly divided into three sets. The first one is the training. The second set is for validation. The last set is for testing. In ML, data splitting is usually divided into 60-20-20. This means that 60 percent of the data is used in the training, 20 percent is used for validation and 20 percent for testing. The training set is used to teach the model where the model learns to associate the features with the final label. The validation set is used to fine tune the model parameter after every epoch and check whether the model is performing better or not aiming to improve the model's performance. The test set is used after the training and validation are done, to make sure that the model can perform well with new unseen data.

2.1.2 CNN Architecture

2.1.2.1 Neural networks

Neurons are the primary unit of processing in NN (O'Shea & Nash, 2015). Fig.2.5 shows the neurons general contents. Where X represent the inputs, W signify the weights, bias, activation function, and Y indicate the output. Where $(x_i, W, b) = \sigma(Wx_i + b)$. Weights and biases are considered as learnable parameters in NN.

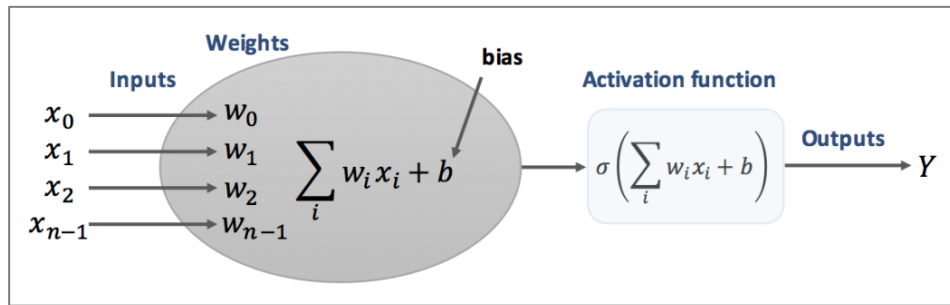


Fig.2.5 Mathematical representation of a neuron (Liu, 2017).

NN is built from one or many fully connected layers also known as a dense layer. The dense layer is built from numerous neurons. Every neuron is connected to every neuron in the prior layer as shown in Fig.2.6. Each neuron's function in the same layer is completely independent of any other neuron. The dense layer has two hyperparameters: the number of neurons and the activation function. The number of neurons is affected by the solution complexity. As the solution complexity increases the neuron number increase. The last dense layer in NN is called the output layer and it denotes the class score. An artificial neural network obtains in an input, then that input proceeds through the network then the neural network produces an output vector. Using Fig.2.6, identify the activation output of the j^{th} neuron in the i^{th} layer as a_j^i , where a_j^i is the j^{th} aspect in the input vector.

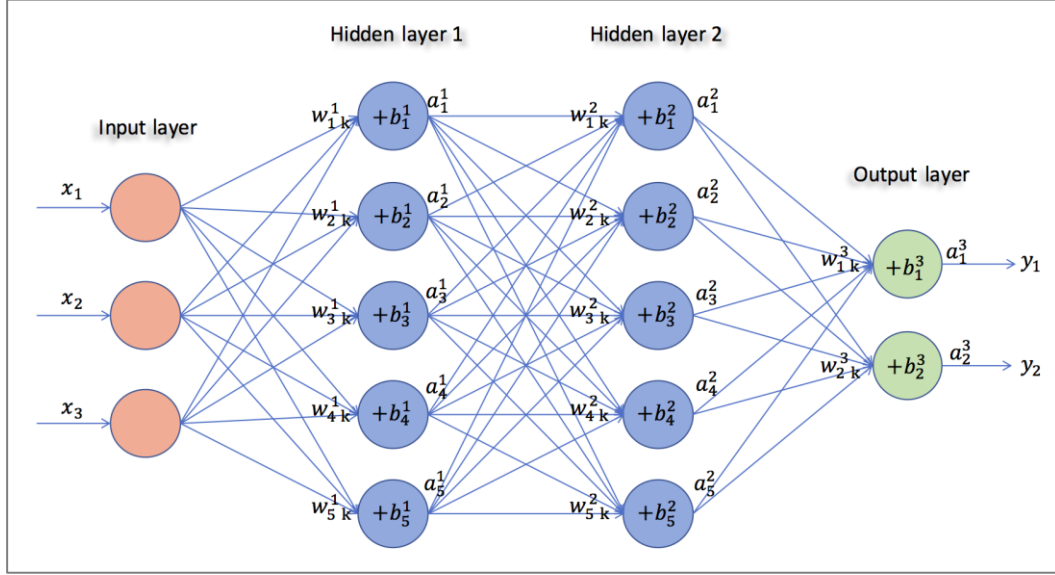


Fig.2.6 3-layer artificial neural network with two hidden layers (Liu, 2017)

Next layer input can be calculated using equation (2.1):

$$a_j^i = \sigma(\sum_k w_{jk}^i a_k^{i-1} + b_j^i) \quad (2.1)$$

Where symbols in equation (2.1) are:

- Activation function is donated as σ .
- w_{jk}^i represents the j^{th} neurons weight from the $(i - 1)^{th}$ layer to the j^{th} neuron in the i^{th} layer.
- The bias from the i^{th} layer in the j^{th} neuron is indicated as b_j^i .
- The activation value from the i^{th} layer in the j^{th} neuron is donated as a_j^i .

Sometime, $\sum_k w_{jk}^i a_k^{i-1} + b_j^i$ is written as z_j^i , which represent the activation value of a neuron prior than operating the activation function as represented in equation (2.2).

$$z_j^i = \sum_k w_{jk}^i a_k^{i-1} + b_j^i \quad (2.2)$$

2.1.2.2 Convolution Neural Network

CNN is one of the DL algorithms that can recognize and classify features. CNN represents a special class of DL and has become a standard tool in computer vision applications. CNN architecture can be divided into two parts as seen in Fig.2.8. Convolution part where FE is done. In addition to, the NN part where classification or prediction is done. The convolution part is built from multiple convolution and pooling layers. Convolutional layer has two main hyperparameters: filter size and activation function. Filter purpose in a convolutional layer is mining the features from the data input. While activation function is used to convert the feature, a map is generated by the convolutional layer to create an activation map. On the other hand, NN part is built from one or many fully connected layers also known as a dense layer. The dense layer is built from number of neurons. Neurons are the primary unit of processing in NN. Neurons have two learnable parameters: weights and biases. The learnable parameters are tracked and adjusted using backpropagation. Every neuron is fully connected to the previous neural layer. Each neuron's function is completely independent of any other neuron. Dense layer has two hyperparameters: the number of neurons and the activation function. The number of neurons increases when the solution complexity increases. The functionality of the activation function in NN is different from the one in the convolutional as it decides whether to activate the neuron or not.

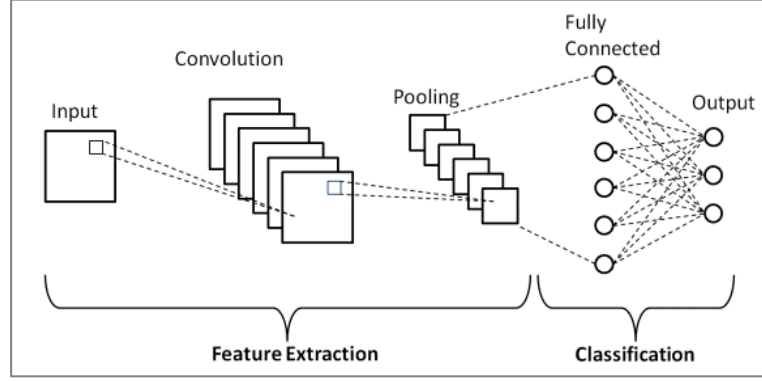


Fig.2.7 Representation of basic CNN architecture (Balaji, 2020).

2.1.2.2.1 Convolutional layer

The Convolutional layer is the CNN's main structural component. It performs the convolution operation using an input image with convolution matrices (kernel filters) to produce the feature maps as an output. Figure 2.9 shows a general convolution operation where $(N \times H \times W)$ is a 3-dimensional matrix input feature map (X) , where number of channels, height and width of the input feature map are symbolized, respectively. On the other hand, a 4-dimensional matrix with $(M \times N \times H_K \times W_K)$ represents the convolution kernel K , where number of filters, number, height, and width of kernels are denoted, respectively. A 1-dimensional matrix with $(M \times 1)$ indicated the bias (b) . Finally, the feature map (O) is the output with $(M \times H_O \times W_O)$ matrix, like the input matrix (M) represents the number of channels in the output feature map, and (H_O) and (W_O) represents the feature map's height and width, respectively. The feature map can be calculated using equation (2.7).

$$O_{i'j'}^m = \sum_{n=2}^N \sum_{i=1}^{H_K} \sum_{j=1}^{W_K} X_{n(i+(i'-1)H_s)(j+(j'-1)W_s)}^m \cdot K_{nij}^m + b_m \quad (2.7)$$

Where $i' \in [1, H_o], j' \in [1, W_o], m' \in [1, M_o]$

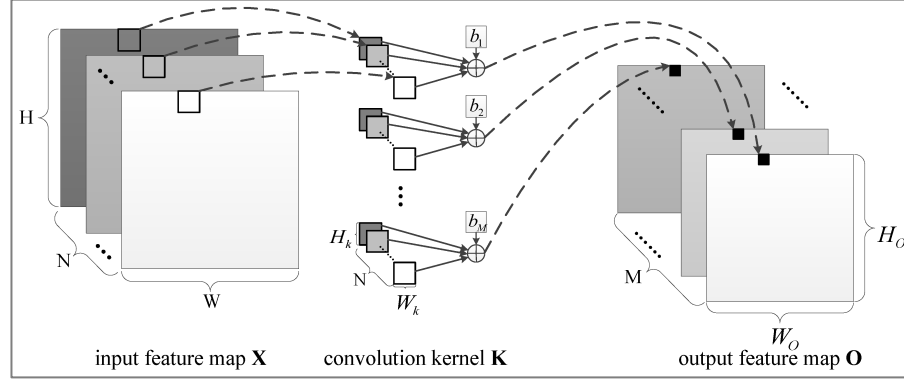


Fig.2.8 Representation of general convolutional operation (Li , Zhou , Wang, Shen, & Ran, 2021).

2.1.2.2.2 Pooling layer

Pooling layer (subsampling layer) is an approach used to down-sample the input size. It is used to decrease the number of parameters in the network which also reduce the computation cost. Moreover, it is used to lower the chance of overfitting. Pooling layers are usually used between two convolutional layers in a CNN model. There are several pooling techniques, including L2-norm, average, and maximum pooling. These methods were used often historically. But recently most of the models used max pooling as works better than the other method in practice as shown in (Li, Karpathy, & Johnson, 2015). As the most used pooling method in image processing tasks is the max pooling, the max pooling method was selected to be used in this work. Max pooling works by cutting the dimension of the input size by selecting the pixel with the maximum value from every set region in an image. An example of max pooling is presented in Fig.2.10.

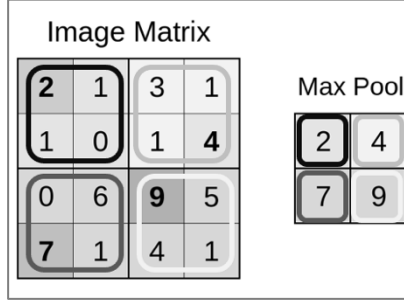


Fig.2.9 Maxpooling example.

2.1.2.2.3 Dropout layer

In this work, dropout layers were added to prevent over-fitting problems. Dropout layers work by randomly setting the activations in neurons to zero in that layer. By doing so, the dropout layer forces the CNN to learn the input characteristics and provide the right output even when some of the neurons are dropped out. Fig.2.11 represent a NN with and without dropout.

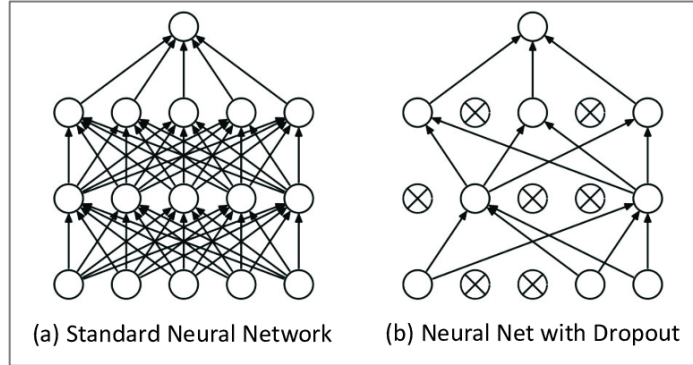


Fig.2.10 NN with and without dropout (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014).

2.1.2.2.4 CNN Hyperparameters

Hyper-parameter optimization is the process of adjusting parameters for a neural network to reach to the desired performance. A brief list of hyper-parameters used in this work is listed in this section:

- **Training epochs:** It is the number of complete passes through the training dataset.

- **Learning rate:** It is used to define the step size between each epoch while aiming toward a minimum of a loss function.
- **Batch Size:** It is the number of samples handled in each training epoch.

2.2 Literature review

A lot of work explores the recognition models for SL worldwide. SL has been approached in two ways. First, the vision-based approach which use cameras to learn and translate. Second, the sensors-based approach which use sensors in glove to learn and translate. This work focuses on the vision-based approach, mainly alphabet. Also, this work concentrates on ArSL recognition using ArSL2018 dataset (Latif, Mohammad, Alghazo, AlKhalaf, & AlKhalaf, 2019). This section is divided into two sub section based on ArSL and non-ArSL. Each subsection starts by presenting models that recognize sign language using ML. After that, the improvement of CNN models in sign language recognition are explored. Lastly, models that applied real time recognition of sign language are stated.

2.2.1 Non-Arabic Sign Language

2.2.1.1 Approaches Based on Traditional Machine Learning

ML has been used to recognize many sign languages, this paragraph proposed some of it like Urdu sign language, Amharic sign language, and Pakistan Sign Language. For Urdu sign language, support vector machine, Random Forest, and K-nearest neighbor ML algorithms where used. It was used in one proposal done by Zahid and his team (Zahid, et al., 2022). They created three models were trained and tested using self-collected dataset. The best performance reached by support vector machine model with 90% accuracy, after that the random forest model with 88% accuracy, and lastly the K-nearest neighbor with the accuracy of 84%. For the Amharic sign language, two models

were implemented, the first one was implemented using a multi-class support vector machine and the second one was implemented using artificial neural network (Tamiru, Tekeba, & Salau, 2022). Both models were trained and tested on self-collected dataset containing the Amharic alphabet. The first one achieved 98.06% accuracy while the second one achieved 80.82% accuracy. For the Pakistan Sign Language, a model done by Shah and his team (Shah, et al., 2021) was implemented using Multiple kernel learning in support vector machine. The model used a static Pakistan sign language. An accuracy of 91.93% was achieved using the proposed model.

2.2.1.2 Approaches Based on Convolutional Machine Learning

Several studies related to non-Arabic sign language have been done, some of these are given here. First, the work of Rahman and his coauthors (Rahman, Islam, Rahman, Rivolta, & Aktaruzzaman, 2019) proposed a CNN model with that has been tested in four ASL datasets. The proposed CNN model has been tested on four datasets, out of which, the model achieved 99%, whereas using the fourth dataset, the model achieved 100%. On the other hand, a CNN model was developed by Jiang, Lu, and Wang (Jiang, Lu, & Wang, 2020) to recognize Chinese sign language. They used a Chinese finger sign language dataset. The model's performance reached to accuracy of 90.91%. An Indian Sign Language (ISL) recognition model was implemented by Sharma and Singh (Sharma & Singh, 2022) using CNN. The model was trained and tested using three datasets. The model achieved an accuracy of 88.01% to 99.52% depending on the datasets. In addition to that, another model was proposed on ASL for Sign language recognition (Obi, Claudio, Budiman, Achmad, & Kurniawan, 2023). It was

implemented using CNN with the ASL dataset. The model achieved 96.3% accuracy using that dataset.

2.2.1.3 Real time recognition

The real time recognition helps the most, as it can be used in direct communication. A real time recognition for non- ArSL was investigated more than the Arabic one, some of the work of non- ArSL are discussed. First, the work of Shenoy and his team (Shenoy, Dastane, Rao, & Vyavaharkar, 2018) proposed a real time recognition tool using K-nearest neighbor. Self-collected dataset was used to train and test both models. The model performance reached an accuracy of 97.2%. In addition to that, an ensemble of three real time recognition models for ASL were implemented by Cayamcela and Lim (Cayamcela & Lim, 2019). The first model is designed form scratch using CNN, the other two were designed using LeNet-5 and MobileNetV2 models. The three models were ensembled and trained and tested using ASL dataset. The ensemble model reached an accuracy of 99.8%. Furthermore, a real time gesture recognition system was implemented using CNN (Kodandaram, Kumar, & L, 2021). The dataset for this model was self-collected by the authors. The proposed model achieved an accuracy of 99.8 %. Finally, a CNN model that recognize ASL in real time was implemented by Taskiran, Killioglu, and Kahraman (Taskiran, Killioglu, & Kahraman, 2018). The ASL dataset was used to train and test the implemented model and achieved the accuracy of 98.05%.

2.2.2 Arabic Sign Language

2.2.2.1 Approaches Based on Traditional Machine Learning

To the best of the researcher 's knowledge, only one work used ArSL2018 dataset with traditional ML techniques. The work was done by

Latif and his team (Latif, Alghazo, Mohammad, & Alghazo, 2021). They proposed an AI translator that translate ArSL using five different ML algorithms. Regression, random tree, bagging classifier, random forest, and bagging classifier were all implemented. ArSL2018 was used to train the algorithms. With an F1-score score of 96.58%, the random Forest algorithm had the best performance.

2.2.2.2 Approaches Based on Convolutional Neural Networks

Many studies have been conducted using CNN with ArSL2018 model. We state the works here below based on the model performance, from lowest to highest. First, Elsayed and Fathy (Elsayed & Fathy, 2020) developed a translator using ontology with CNN. ArSL2018 was used to train the system. The suggested method reached 88.87% accuracy. Second, the work of Zakariah and his team (Zakariah, Alotaibi, Guo, & Mamun Elahi, 2022) got a better performance by comparing six CNN models. They used ArSL2018 to train and compare VGG16 model, ResNet50 model, Xception, VGG16 model, InceptionV3 model, ResNet50 model, MobileNet model and EfficientNetB4 model. The EfficientNetB4 model performed the best, with 95% accuracy. Third, Two CNN models were implmeneteed by Alshomrani, Aljoudi, and Arif (Alshomrani, Aljoudi, & Arif, 2021). The first model has three-layers and the other one has two layers. ArSL2018 dataset was used to train and evaluated both models. The three-layered CNN model achieved better performance with a 96.6% accuracy score. ResNet50, and MobileNetV2 were trained by Alnuaim and his coauthors (Alnuaim, et al., 2022) as hand gesture recognition tools. ArSL2018 was used to train and test the model. The model has a 97% accuracy rate. In Latif and his team (Latif, et al., 2020) work, a CNN architecture was created to recognize ArSL alphabet signs. ArSL2018

was used to train and test the model. The model Obtained an accuracy of 97.6%. Finally, two fine-tuned CNN models were built by Saleh and Issa (Saleh & Issa, 2020). The two models they matched are VGG16 and ResNet152 models. They matched them to create an ArSL recognition tool. This work was trained and tested using ArSL. The proposed models reached an accuracy near 99%.

2.2.2.3 Real time recognition

Recently, real time recognition for sign language started to gain focus. To the knowledge of the researcher, there are only four research works related to ArSL. First, the work of Albelwi and Alginahi (Albelwi & Alginahi, 2012) offered the first model that translates ArSL in real time. They used a ML algorithm K Nearest Neighbor and obtained an accuracy of 90.55%. Second, using a deep learning model, Kamruzzaman (Kamruzzaman, 2020) created a system that speaks out the result in Arabic and detects hand sign letters automatically. This system reached 90% accuracy in recognizing Arabic hand sign-based letters. Third, a CNN model was designed by Alsaadi et.al, (Alsaadi, Alshamani, Alrashdi, Albelwi, & Elfaki, 2022). The model was designed to perform real time recognition. The model achieved 94.81% accuracy. Finally, a work done by Almana and Al-Omary (Almana & Al-Omary, 2022) presented a CNN model that translates ArSL in real time. The model proposed got 95% accuracy and 95% F1-score.

After investigating in ArSL and non-ArSL recognition models, it was concluded that there is not enough investigation in the recognition of ArSL compared to the to the non-ArSL. Although the number of ArSL is increasing slowly, there is still a gap for improving the performance and trying several other models to enhance the accuracy of real time recognition. This work

aimed to have a Real Time Translation of Arabic Sign Language to Arabic Text Using Deep Learning Model that have better performance than existing model.

Methodology

3 Methodology

This chapter is devoted on the used dataset and the design of the adopted CNN model. In addition, the model evaluation metrics are stated. The first section of this chapter presents a detailed description of the dataset. Second, we explain the proposed model design and illustration. In the final section, a description of the evaluation metrics is presented.

3.1 The Proposed Architecture

The proposed application aimed to translates ArSL signs to written letters in real time. It was implemented after finishing three parts to achieve the application's aim as Fig.3.1 presents. First, we started to train the model using the CNN proposed architecture. Then, the model is saved and evaluated. Finally, the real time recognition part. This section elaborated on the implementation of the two parts.

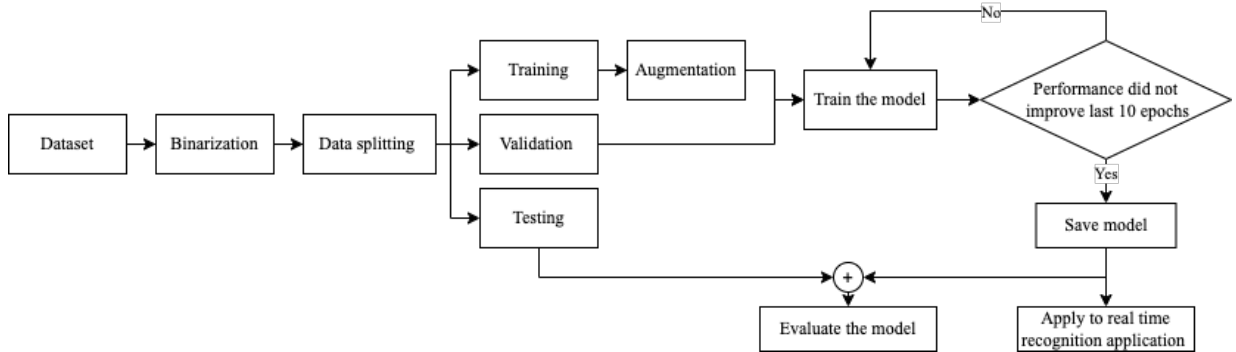


Fig.3.1 Architecture for the proposed approach

3.1.1 CNN Proposed Architecture

CNN architecture was chosen to implement this model as it is the best NN for image processing. The CNN architecture for the translator model is illustrated in Fig.3.2. It consisted of four convolutional layers, four pooling layers, three dropout layers, a flatten layer and four dense layers.

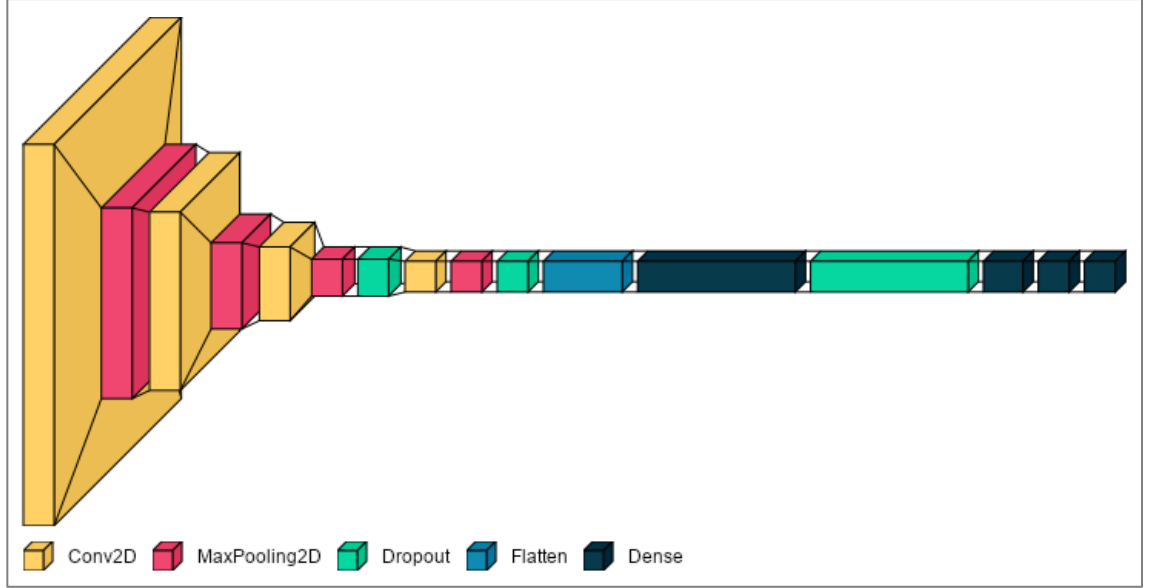


Fig.3.2 Illustration of proposed CNN model.

All the convolutional layers used in this architecture have the same activation function ReLU and 3×3 kernel size. The ReLU activation function is used because of two reasons: First it is a simple math operation which make it computationally inexpensive. Second, it is immunized from the vanishing gradient problem. Additionally, the pooling layers have the same kernel size 2×2 . Moreover, the dropout rate in all the dropout layers is 20%. Table 3.1 presented each layer and its type, its configuration, and the number of its parameters. The input layer is the first layer, it is a binary image sized 64×64 pixels. Then from the second to the seventh layers are three sets of convolutional and pooling layer. The second layer is a convolutional layer that has 16 filters. The next layer is a max pooling layer. The fourth layer is another convolutional layer that has 32 filters. The fifth layer is a max pooling layer. The sixth layer is a convolutional layer that has 64 filters. The seventh layer is a max pooling layer. For the eighth layer, a dropout layer is used as regularize. Then the ninth and tenth layers are convolutional layer that has

128 filters and a max pooling layer, respectively. The eleventh layer is dropout layer. The twelfth layer is the flatten layer that convert the data from 2-dimanectional data into a vector. The thirteenth layer is a fully connected layer, it has 1024 neurons and its activation function is the ReLU. Followed by a dropout layer. Then two additional dense layers one with 256 neurons and 64 neurons both activated by ReLU. Finally, the output layer is a dense layer with 32 neurons representing each class. It is activated using Softmax to classify the output. The filters number are increasing in the convolutional layers as the network deepens, this is done because the information in the prior layers is noisy, so while the information gets clearer the filter size increases to collect the most combination of the clean information. The model parameter was optimized using Adam optimizer. Additionally, the categorical cross-entropy function was used to calculate the loss. Additionally, the checkpoint method was employed to store the performance-optimized epoch. Every epoch, Checkpoint checks the recall validation value. The checkpoint stores the epoch when the recall validation value increases. Additionally, early stop was employed to interrupt training when the model's performance stopped advancing across epochs. Early stop has a parameter called patience. patience stores a number value then track how many epochs did not improve. It stops when the number of epochs that did not improve reach the patience value. In this work, patience was set as 4. The batch size for training in the proposed model was set as 150. Additionally, learning epochs were set at 150 epochs. In addition to that, early stop and checkpoint were set with patience as 10 tracking validation recall.

Table 3.1 The configuration and the number of parameters in the proposed architecture

#	Layer	Layer Configuration	# Parameter
1	Input	1×64×64	0
2	Convolution	16 filter, 3×3 kernel, fctivation function = ReLU	160
3	Max pooling	pool_size=2×2	0
4	Convolution	32 filter, 3×3 kernel, fctivation function = ReLU	4,640
5	Max pooling	pool_size=2×2	0
6	Convolution	64 filter, 3×3 kernel, fctivation function = ReLU	18,496
7	Max pooling	pool_size=2×2	0
8	Dropout	Dropout property= 20%	0
9	Convolution	128 filter, 3×3 kernel, fctivation function = ReLU	73,856
10	Max pooling	pool_size=2×2	0
11	Dropout	Dropout property= 20%	0
12	Flatten	-	0
13	Dense	1024 Neurons, fctivation function = ReLU	525,312
14	Dropout	Dropout property= 20%	0
15	Dense	256 Neurons, fctivation function = ReLU	262,400
16	Dense	64 Neurons, AF = ReLU	16,448
17	Dense (Output)	32 Neurons, AF = Softmax	2,080
Total number of parameters		903,392	

3.2 Real Time implementation

The real time recognition in the proposed model is applied with the help of OpenCV. The real time recognition workflow is displayed in Fig.3.3. OpenCV in the presented workflow captured real time images using the computer's camera. We started by creating a dialog that displays the real time video is shown in the screen. Then, to mimic a mirror and to make the left hand appears on the left and the right hand appears on the right the frame is flipped. Next, a square specified in the dialog shows where the Region of Interest (ROI) is. Now the dialog is ready to capture the images. After the image is captured, the ROI is cropped and the preprocessing that was done to the training is done to it. Starting up with resizing the ROI to 64×64. Then grey scaling and binarization is done to the ROI using the same threshold used

in the training data. After the preprocessing, the ROI is served to the pretrained model to start the predication. Finally, the prediction by the model is displayed in the same dialog.

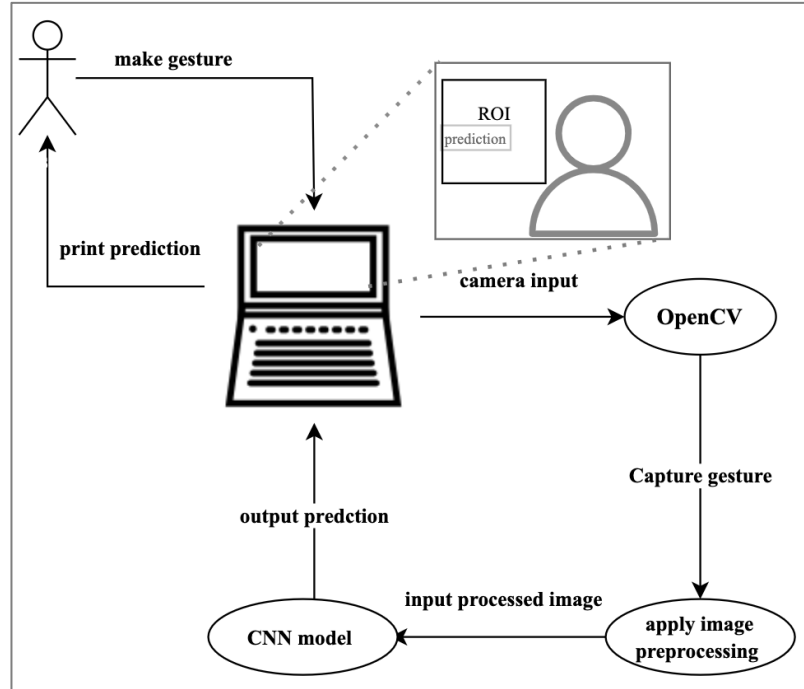


Fig.3.3 Real time workflow

Experimentations and Results

4 Experimentations and Results.

In this chapter, two subsections are stated. The first section of this chapter describes the experimentation. The second section presents Results.

4.1 Dataset

4.1.1 Dataset description

ArSL2018 dataset was used as the learning field in this work. It is a dataset published by Prince Mohammad bin Fahd University (Latif, Mohammad, Alghazo, AlKhalaf, & AlKhalaf, 2019) for the public with the aim of helping in ML research. The dataset is consisted of 54049 images captured using smart camera. The dataset is classified into 28 Arabic alphabet characters' signs and four heavily used combination of Arabic alphabet characters. The characters' signs were signed by over 40 volunteers. Volunteers were standing around one meter away from the camera. The images were captured with different angles, backgrounds and lighting to insure the variety of hand sizes and environment. There are different number of images for every alphabet in the dataset is ranged between 1300 to 2000 images. The camera captured the images in RGB. The data collector did some pre-processing by converting the images to grayscale and fixed the dimension to 64 by 64 to make the images appropriate for recognition and classification. Each character, its label in the dataset, its gesture and how many images of the character in the dataset is shown in table 4.1.

Table 4.1 Data, Gesture and Number of Images

#	Character In Data	Gesture	# Images	#	Character In Data	Gesture	# Images
1	ع Ain		2,114	17	ل Laam		1,832
2	ا Al		1,343	18	م Meem		1,765

3	أ ALEFF		1,672	19	ن Nun		1,819
4	ب Bb		1,791	20	ر Ra		1,659
5	د Dal		1,634	21	ص Saad		1,895
6	ذ DHA		1,723	22	س Seen		1,638
7	ض Dhad		1,670	23	ش Sheen		1,507
8	ف Fa		1,955	24	ط Ta		1,816
9	ق Gaaf		1,705	25	ت Taa		1,838
10	غ Ghain		1,977	26	ث Thaa		1,766
11	ه Ha		1,592	27	ذ Thal		1,582
12	ح Haa		1,526	28	ة Toot		1,791
13	ج Jeem		1,552	29	و Waw		1,371
14	ك Kaaf		1,774	30	ي Ya		1,722
15	خ Khaa		1,607	31	پ Yaa		1,293
16	لا La		1,746	32	ز Zay		1,374

4.1.2 Data Preprocessing

For every image in the dataset three preprocess was applied. First, the data was sorted by category. Every character represents a category and was stored in separated file. Then, the image was converted to a binary image using binarization method with threshold value 150. After that, the data was divided into training, validation and testing using data splitting method with 60% for training, 20% validation, and 20% for testing. This splitting approach was selected because it allowed to fine tune the model parameter after every epoch using the validation set to reach to the best performance. Also, the testing set

allowed us to make sure that the model does not suffer from overfitting by testing it with new data.

4.2 Evaluation Metrics

Recall, precision, F1-score, accuracy, weighted-average F1-score, and macro-averaged F1-score were used to evaluate the proposed model. The evaluation is done after it was successfully trained and tested using the ArSL dataset2018. To calculate the evaluation metrics, confusion matrix table is used. Confusion matrix is a matrix table where the first axis is predictive, and the second axis is the actual values. Table 4.2 represent the confusion matrix. It has four results True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN).

Table 4.2 Confusion Matrix

		Predictive values	
		False	True
Actual values	False	TN	FP
	True	FN	TP

Binary classification can be evaluated using four measures: recall, precision, F1-scores and accuracy which are calculated using the confusion matrix. Equations (4.1), (4.2), (4.3), and (4.4) shows how it can be calculated.

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4.1)$$

$$\text{Precision} = TP/(TP + FP) \quad (4.2)$$

$$\text{F1 - scores} = 2 \frac{\text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}} \quad (4.3)$$

$$\text{Accuracy} = \frac{TN+TP}{TN+TP+FN+FP} \quad (4.4)$$

On the other hand, for multiclassification each class's recall precision, F1-score and accuracy need to be calculated separately. The equations used to calculate each class's recall, precision, F1-scores and accuracy are illustrated in Equations (4.5), (4.6), (4.7) and (4.8), respectively.

$$\text{Recall}_c = \frac{TP_c}{TP_c + FN_c} \quad (4.5)$$

$$\text{Precision}_c = \frac{TP_c}{TP_c + FP_c} \quad (4.6)$$

$$\text{F1-score}_c = 2 \frac{\text{Recall}_c \times \text{Precision}_c}{\text{Recall}_c + \text{Precision}_c} \quad (4.7)$$

$$\text{Accuracy}_c = \frac{TN_c + TP_c}{TN_c + TP_c + FN_c + FP_c} \quad (4.8)$$

The small c in equations (4.5), (4.6), (4.7) and (4.8) represents the calculated class. After that, formula (4.9) was used to calculate macro-average F1-score. Macro-averaged F1-score is considered one-vs-rest F1-score averaging approach because it works by calculating the arithmetic mean for one class against all other classes as explained by Takahashi and his team (Takahashi, Yamamoto, Kuchiba, & Koyama, 2022).

$$\text{Macro averaged} = \frac{\text{F1-scores}_{c_1} + \text{F1-scores}_{c_2} + \dots + \text{F1-scores}_{c_n}}{n} \quad (4.9)$$

Next, Formula (4.10) was used to calculate the weighted average F1-score. Weighted-average F1-score is another one-vs-rest F1-score averaging approach. It is computed by calculating the mean of every class's F1-scores with respect to each class's actual occurrences.

$$\begin{aligned} \text{Weighted Averaged} = & (\text{F1-scores}_{c_1} \times \\ & \text{support proportion}_{c_1}) + (\text{F1-scores}_{c_2} \times \\ & \text{support proportion}_{c_2}) + \dots + (\text{F1-scores}_{c_n} \times \\ & \text{support proportion}_{c_n}) \end{aligned} \quad (4.10)$$

4.3 Experimentations

4.3.1 Development Requirements

4.3.1.1 Hardware requirements

CNN is a field of ML that needs powerful computational requirements, so CNN usually is implemented in graphics processing units (GPUs) to speed up all computation work as it is computationally powerful, because without GPUs a model could take weeks or even months to finish an experiment or execute an experiment for a 12 hours or more and end up finding out that the chosen parameters were unfitting. In this work, we used two GPUs Intel UHD Graphics 630 and Radeon Pro 555X with 16 GB RAM and 6-Core Intel Core i7 to train and test the translator tool.

4.3.1.2 Software requirements

4.3.1.2.1 TensorFlow

TensorFlow is an open-source Python library that is used for fast numerical computing (Abadi, et al., 2016). TensorFlow was created by Google and released inside Apache 2.0 open-source license. It is one of the foundation libraries used to create deep learning models. Some time it is used with other wrapper libraries like Keras which simplify the process built on top of TensorFlow. TensorFlow can function on mobile devices, GPUs as well as single CPU systems also it can function on large scale distributed systems.

4.3.1.2.2 Keras

Keras is an open-source API built in Python which uses either Theano or TensorFlow as backend. It was developed focusing on supporting fast experimentation, developers made it is easier to build whole solutions, and

easy to read with the best selection of state-of-the-art algorithms like optimizers, normalization routines, and activation functions.

4.3.1.2.3 OpenCV

OpenCV is an open-source software library used for computer vision. To deliver a common infrastructure for computer vision applications and to fast-track the use of machine point view in the commercial products OpenCV was developed. OpenCV makes it easy for businesses to operate and alter the code because it is one of the Apache 2 licensed product.

4.4 Results

4.4.1 Application performance

This section presented the evaluations and results after training the model 7 times to demonstrate the consistency of the model's performance. Fig.4.1 presents the training and validation evaluation result for the fastest training trail. Fig.4.1 showed that early stop interrupted the model training at the 35th epoch which means that checkpoint was used to save the parameters of the 25th epoch. The saved trained model got the loss as 0.2127, precision as 94.91% recall as 92.10%, and accuracy as 93.28%. According to the validation findings, at the 25th learning epoch, validation precision, recall, and accuracy all achieved 97.78%, 96.59% and 96.82%, respectively with a validation loss as low as 0.1265. On the other hand, Fig4.2 presents the training and validation evaluation result for the slowest training trail. Fig.4.2 presents that early stop interrupted the model training at the 76th epoch which means that checkpoint was used to save the parameters of the 66th epoch. The saved trained model got the loss as 0.1071, precision as 97.29% recall as 96.15%, and accuracy as 96.63%. According to the validation findings, at the 25th learning epoch, validation precision, recall, and accuracy all achieved

98.10%, 97.33% and 97.44%, respectively with a validation loss as low as 0.1143.

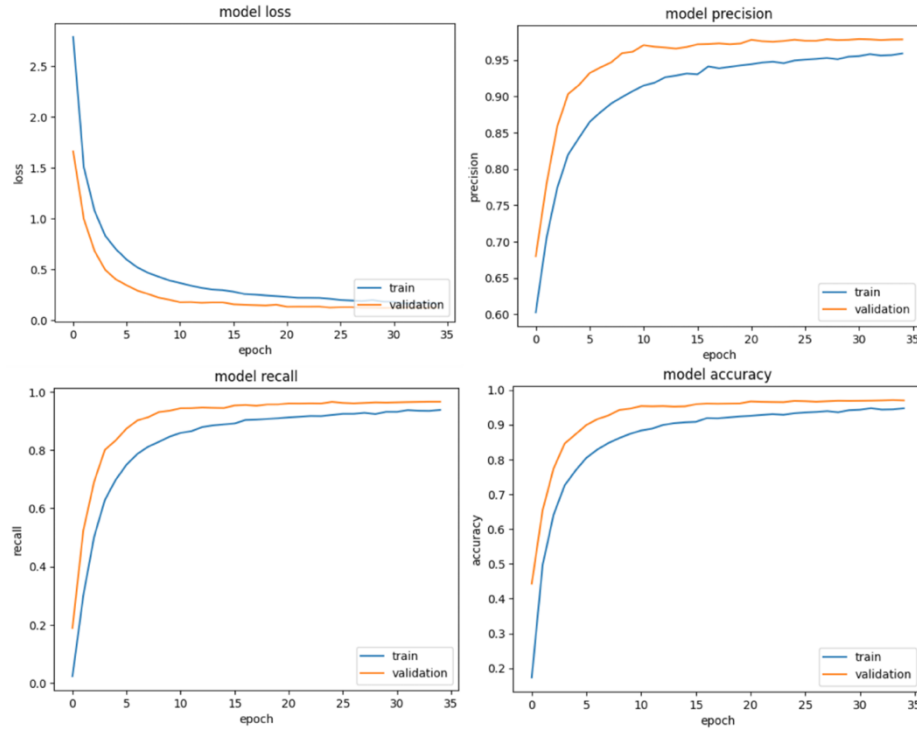


Figure 4.1 illustration of evaluation measurements for the training and validation of the fastest trail mode.

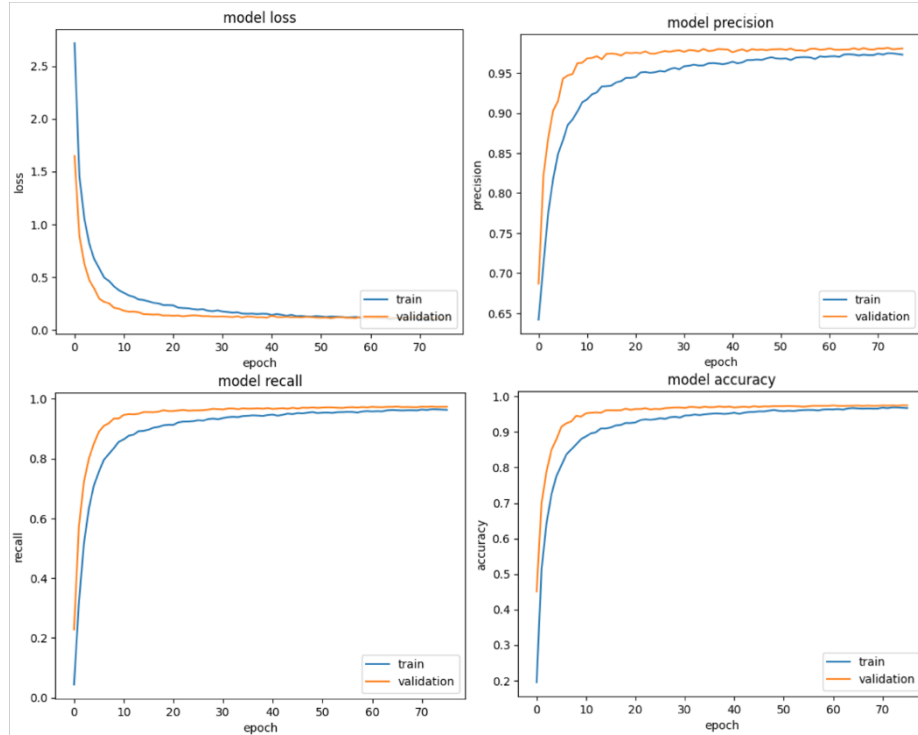


Figure 4.2 illustration of evaluation measurements for the training and validation of the slowest trail mode.

The average of the performance of the 7 trail of the model is presented in Table 4.1. The classification report's components are each class's evaluation measure and its support. Along with the multiclass' accuracy, the weighted average, and the macro average. Table 4.1 shows an average classification report of the 7 trails of the proposed model. Table 4.1 shows that Yaa and La performed highest in all performance classes, with 99% F1-score, 100% recall, and 99% precision for Yaa and 98% F1-score, 99% recall, and 99% precision for La. The worst performance class was Ha and Gaaf with 93% precision, 96% recall, and 95% F1-score for Ha and 95% precision, 95% recall, and 95% F1-score for Gaaf. For the F1-score classes accuracy 97% was achieved. Similarly, both macro average and weighted average performed 97%.

Table 4.3 Output of the classification report for the testing

#	Class	Precision	Recall	F1-score	Support
1	Ain	97%	99%	98%	424
2	Al	99%	98%	98%	270
3	ALEFF	97%	97%	97%	335
4	Bb	97%	99%	98%	359
5	Dal	96%	97%	97%	328
6	DHA	96%	96%	96%	346
7	Dhad	98%	99%	98%	334
8	Fa	95%	96%	96%	391
9	Gaaf	95%	95%	95%	341
10	Ghain	99%	98%	98%	396
11	Ha	93%	96%	95%	319
12	Haa	96%	95%	96%	306
13	Jeem	96%	94%	95%	311
14	Kaaf	97%	97%	97%	356
15	Khaa	97%	95%	96%	322
16	La	98%	99%	99%	350
17	Laam	99%	97%	98%	367
18	Meem	99%	96%	97%	353
19	Nun	99%	98%	98%	365
20	Ra	96%	97%	96%	333
21	Saad	97%	96%	97%	379
22	Seen	98%	98%	98%	329
23	Sheen	97%	98%	98%	302
24	Ta	95%	95%	95%	364
25	Taa	97%	97%	97%	369
26	Thaa	96%	96%	96%	354
27	Thal	97%	97%	97%	317
28	Toot	97%	97%	97%	359
29	Waw	97%	97%	97%	275
30	Ya	98%	99%	99%	345
31	Yaa	99%	100%	99%	260
32	Zay	97%	94%	96%	276
accuracy		97%	97%	97%	10835
macro avg		97%	97%	97%	10835
weighted avg		97%	97%	97%	10835

4.4.2 Real Time Application

Next, the real time application was applied. Using the computer laptop and OpenCV the real time capture frames in real time and feed it to the pertain and pretested model. Finally, the prediction of the model is displayed in the dialog. Fig.4.2 shows an example where the user is signing the sign letter Aleef and the prediction is displayed as text in the dialog. Real time recognition testing has many challenges and it cost a lot as the testing needs to be done by many people with different hand size, also the testing needs to be do in veriaty of backgrounds and lighting. As there was no accessible way to test the real time recognition, and there was no real time recognition in any of the previous work. I was not able to test the real time recognition .



Fig.4.3 A try out of the final application result.

4.4.3 Discussion

After the training and testing is done, we noticed from Fig.4.1 that the validation of the training is better than the testing. This is normal behavior when using drop out. Drop out stops some of neuron from firing in the training. On the other hand, in the validation every neuron is activated and the

whole CNN model is tested. This gives the model more information about the data that it was trained on. Additionally, in Table 4.1, we found that the worst sign performance was Ha and this was expected because the hand sign for Ha and Gaaf is similar as Fig.4.3 showed.

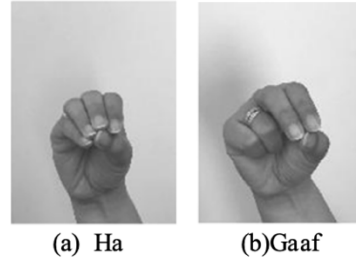


Figure 4.4 (a) Ha sign (b) Gaaf sign

Referencing to Table 4.1, comparing the performance of the previous work using ArSL and the presented CNN model, we found that the presented model has a similar F1-score compared to the F1-score that obtained in paper number 1 and 2 from Table 4.2, but our system different from these papers as our system support the recognition in real-time translating while other do not. On the other hand, paper at 4 and 5 implements real-time recognition with an F1-score of 90% and 95%, respectively. Table 4.2 shows that the proposed system achieved better F1-score compared to with the system introduced in paper number 4 and 5 shows that our system gives a better F1-score. Getting the best F1-score in real-time is a challenging problem and the proposed system achieved the objectives of the research: First, a real time ArSL recognition application was implemented successfully. Second, created a tool that will improve the communication between hearing impaired and hearing individuals Finally, the performance of currently available models is better than the other as Fig.4.4 shows.

Table 4.4 Models Comparison

#	Paper	F1-score	Model	Dataset	Support real-time
1	(Zakariah, Alotaibi, Guo, & Mamun Elahi, 2022)	0.97	CNN	ArSL2018	Do not support
2	(Alshomrani, Aljoudi, & Arif, 2021)	0.97	CNN	ArSL2018	Do not support
3	(Latif, Alghazo, Mohammad, & Alghazo, 2021)	0.90	CNN	ArSL2018	Do not support
4	(Kamruzzaman, 2020)	0.94	CNN	ArSL2018	Do support
5	(Almana & Al-Omary, 2022)	0.95	CNN	ArSL2018	Do support
6	The proposed model	0.97	CNN	ArSL2018	Do support

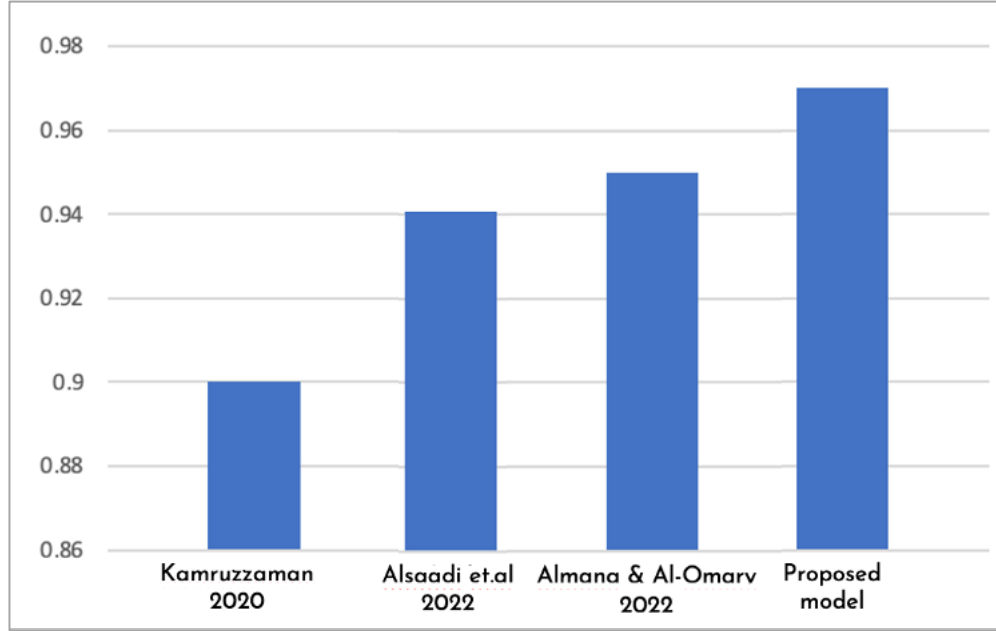


Fig.4.5 Compression of the three-model supporting real time from Table 4.2

Conclusion and Recommendation

5 Conclusion and Recommendation

This chapter presents the conclusion in first section. On the second section, recommendation is stated.

5.1 Conclusion

This work proposed an application that translate Arabic sign language to Arabic text using deep learning. ArSL2018 dataset was used to train the CNN model, with over 50000 images divided into 32 categories. Next, the real time was implemented using OpenCV to capture the real-time frames. After that, the frames from the OpenCV are feed to the CNN pretrained model to recognize the 32 signs. The result showed the model reached weighted-average F1-score of 97%, similar to the macro-average F1-score and accuracy. Which is the best model compared to all of the previous work as Table 4.2 presents. Finally, the tool was able to run and predict the sign in real time successfully. Which is an important feature that is not supported by other similar systems with this performance measure. This application is a great step in improving the real time recognition in the future.

5.2 Recommendation

We recommend that more work of AI and ML be done to help disabled individuals, especially deaf people. For the future work, a new feature can be used to save the spelling of the sign and display a whole word. This new feature will help deaf people to spell words for hearing people easily.

References

- Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., . . . others. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.
- Abdel-Fattah, M. A. (2005). Arabic sign language: a perspective. *Journal of deaf studies and deaf education*.
- Agarap, A. F. (2018). Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*.
- Albelwi, N. R., & Alginahi, Y. M. (2012). Real-time arabic sign language (arsl) recognition. *International Conference on Communications and Information Technolog*.
- Almana, S., & Al-Omary, A. (2022). Real-time Arabic Sign Language Recognition using CNN and OpenCV. *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 32--36.
- Alnuaim, A., Zakariah, M., Hatamleh, W. A., Tarazi, H., Tripathi, V., & Amoatey, E. T. (2022). Human-Computer Interaction with Hand Gesture Recognition Using ResNet and MobileNet. *Computational Intelligence and Neuroscience*.
- Alsaadi, Z., Alshamani, E. a., Alrashdi, A. A., Albelwi, S., & Elfaki, A. O. (2022). A real time Arabic sign language alphabets (ArSLA) recognition model using deep learning architecture. *Computers*.
- Alshomrani, S., Aljoudi, L., & Arif, M. (2021). Arabic and american sign languages alphabet recognition by convolutional neural network. *Advances in Science and Technology. Research Journal*.

- Balaji, S. (2020). *Binary Image classifier CNN using TensorFlow* . Retrieved from medium: <https://medium.com/techiepedia/binary-image-classifier-cnn-using-tensorflow-a3f5d6746697>
- Cayamcela, M. E., & Lim, W. (2019). Fine-tuning a pre-trained convolutional neural network model to translate American sign language in real-time. *2019 International Conference on Computing, Networking and Communications (ICNC)*, 100--104.
- Chaki, N., Shaikh, S. H., Saeed, K., Chaki, N., Shaikh, S. H., & Saeed, K. (2014). *A comprehensive survey on image binarization techniques*. Springer.
- Dietterich, T. (1995). Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 326--327.
- Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*.
- Elsayed, E. K., & Fathy, D. R. (2020). sign language semantic translation system using ontology and deep learning. *International Journal of Advanced Computer Science and Applications*.
- Fakhry, M., & Brery, A. F. (2022). A comparison study on training optimization algorithms in the biLSTM neural network for classification of PCG signals. *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, 1--6.
- Girosi, F., Jones, M., & Poggio, T. (1995). Regularization theory and neural networks architectures. *Neural computation*, 219--269.
- Guyon, I., Gunn, S., Nikraves, M., & Zadeh, L. A. (2008). *Feature extraction: foundations and applications*. Springer.

- Hao, X., Zhang, G., & Ma, S. (2016). Deep learning. *International Journal of Semantic Computing*, 417--439.
- Hecht-Nielsen, R. (1992). Theory of the backpropagation neural network. *Neural networks for perception*, 65--93.
- Hochreiter, S. (1998). The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 107--116.
- Ismail, M. H., Dawwd, S. A., & Ali, F. H. (2021). Static hand gesture recognition of Arabic sign language by using deep CNNs. *Indonesian Journal of Electrical Engineering and Computer Science*.
- Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. *arXiv preprint arXiv:1702.05659*.
- Jiang, X., Lu, M., & Wang, S.-H. (2020). An eight-layer convolutional neural network with stochastic pooling, batch normalization and dropout for fingerspelling recognition of Chinese sign language. *Multimedia Tools and Applications*, 15697--15715.
- Kamruzzaman, M. (2020). Arabic sign language recognition and generating Arabic speech using convolutional neural network. *Wireless Communications and Mobile Computing*.
- Ketkar, N. (2017). Stochastic gradient descent. *Deep learning with Python: A hands-on introduction*, 113--132.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kodandaram, S. R., Kumar, N. P., & L, S. G. (2021). Sign Language Recognition. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 994--1009.

- Krig, S., & Krig, S. (2016). Image pre-processing. In S. Krig, & S. Krig, *Computer Vision Metrics: Textbook Edition* (pp. 35--74). Springer.
- Latif, G., Alghazo, J., Mohammad, N., & Alghazo, R. (2021). Communicating with the deaf and hard of hearing through automatic arabic sign language translator. *Journal of Physics: Conference Series*, 012055.
- Latif, G., Mohammad, N., Alghazo, J., AlKhalaf, R., & AlKhalaf, R. (2019). ArASL: Arabic alphabets sign language dataset. *Data in brief*, 103777.
- Latif, G., Mohammad, N., AlKhalaf, R., AlKhalaf, R., Alghazo, J., & Khan, M. (2020). An automatic Arabic sign language recognition system based on deep CNN: an assistive system for the deaf and hard of hearing. *International Journal of Computing and Digital Systems*, 715-724.
- LeCun, Y., & Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*.
- Li, J., Zhou, X., Wang, B., Shen, H., & Ran, F. (2021). *Design of Efficient Floating-Point Convolution Module for Embedded System*. Retrieved from mdpi: <https://www.mdpi.com/2079-9292/10/4/467/htm>
- Li, F.-F., Karpathy, A., & Johnson, J. (2015). *Convolutional neural networks for visual recognition*. Retrieved from <http://cs231n.github.io/convolutional-networks>
- Liddell, S. K. (2003). *Grammar, gesture, and meaning in American Sign Language*. Cambridge University Press.
- Liu, Q. (2017). *Deep learning applied to automatic polyp detection in colonoscopy images: master thesis in System Engineering with Embedded Systems*. University College of Southeast Norway, Faculty of Technology and Maritime.

- Mukushev, M., Sabyrov, A., Imashev, A., Koishibay, K., Kimmelman, V., & Sandygulova, A. (2020). Evaluation of manual and non-manual components for sign language recognition. *Proceedings of The 12th Language Resources and Evaluation Conference*. European Language Resources Association (ELRA).
- O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.
- Obi, Y., Claudio, K. S., Budiman, V. M., Achmad, S., & Kurniawan, A. (2023). Sign language recognition system for communicating to people with disabilities. *Procedia Computer Science*, 13-20.
- Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 145--151.
- Rahman, M. M., Islam, M. S., Rahman, M. H., Rivolta, M. W., & Aktaruzzaman, M. (2019). A new benchmark on american sign language recognition using convolutional neural network. *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 1--6.
- Reading the Signs: Diverse Arabic Sign Languages*. (2014, August 6). Retrieved from CSIS: <https://www.csis.org/analysis/reading-signs-diverse-arabic-sign-languages>
- Saleh, Y., & Issa, G. (2020). Arabic sign language recognition through deep neural networks fine-tuning. *International Association of Online Engineering*.
- Sathya, R., & Abraham, A. (2013). Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 34--38.

- saudi sign language association*. (2022). Retrieved from <https://saudisla.org/>
- Shah, F., Shah, M. S., Akram, W., Manzoor, A., Mahmoud, R. O., & Abdelminaam, D. S. (2021). Sign language recognition using multiple kernel learning: A case study of Pakistan sign language. *Ieee Access*, 67548--67558.
- Sharma, S., & Singh, S. (2022). Recognition of Indian sign language (ISL) using deep learning model. *Wireless Personal Communications*, 1--22.
- Shenoy, K., Dastane, T., Rao, V., & Vyavaharkar, D. (2018). Real-time Indian sign language (ISL) recognition. *2018 9th international conference on computing, communication and networking technologies (ICCCNT)*, 1-9.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 1--48.
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 1929--1958.
- Takahashi, K., Yamamoto, K., Kuchiba, A., & Koyama, T. (2022). Confidence interval for micro-averaged F 1 and macro-averaged F 1 scores. *Applied Intelligence*, 4961--4972.
- Tamiru, N. K., Tekeba, M., & Salau, A. O. (2022). Recognition of Amharic sign language with Amharic alphabet signs using ANN and SVM. *The Visual Computer*, 1--16.
- Taskiran, M., Killioglu, M., & Kahraman, N. (2018). A Real-Time System for Recognition of American Sign Language by using Deep Learning. *2018 41st international conference on telecommunications and signal processing (TSP)*, 1--5.

- Zahid, H., Rashid, M., Syed, S. A., Ullah, R., Asif, M., Khan, M., . . . Khan, A. H. (2022). A computer vision-based system for recognition and classification of Urdu sign language dataset. *PeerJ Computer Science*, e1174.
- Zakariah, M., Alotaibi, Y. A., Guo, Y., & Mamun Elahi, M. (2022). Sign language recognition for Arabic alphabets using transfer learning technique. *Computational Intelligence and Neuroscience*.
- Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.
- Zelaya, C. V. (2019). Towards explaining the effects of data preprocessing on machine learning. *2019 IEEE 35th international conference on data engineering (ICDE)*.

Appendix

Appendix

Activation function

The most important part of in neural network is the activation function also known as the transfer function or threshold function. The activation function is used to decide whether the neuron should be activated or not. The most popular activation functions are tanh, rectified linear unit (ReLU), and sigmoid. These are usually used because they make model learning quicker than other functions and they prevent saturation issues. In this work, ReLU and Softmax are used.

ReLU

ReLU function is the most used function in deeper NN architectures (Agarap, 2018). It is commonly used as it is much easier to calculate than other functions as shown in equation (2.3). Additionally, ReLU is less vulnerable to the vanishing gradient problem. Vanishing gradient (Hochreiter, 1998) is a problem that occurs when deep multilayer NN loses the ability to propagate beneficial gradient value from the model output layers back to the layers near the input layers.

$$a_j^i = \sigma(z_j^i) = \max(0, z_j^i) \quad (0.1)$$

Softmax function

Softmax function is mostly used in the output layer in NN (Agarap, 2018). It is the function that generates the class score over (C) classes. The sum of the classes' scores is 1.0, and the class with the highest score is selected as the prediction. Softmax function is defined as in equation (2.4)

$$y_c = \varrho(Z)_c = \frac{e^{z_c}}{\sum_{d=1}^C e^{z_d}} \text{ for } c=1 \dots C \quad (0.2)$$

Softmax ϱ in equation (2.4) takes C-dimensional vector (Z) as an input and yield C-dimensional vector (y) as an output with a real value between 0 and 1. Softmax uses $\sum_{d=1}^C e^{z_d}$ as a regularizer to assure that $\sum_{c=1}^C y_c = 1$.

Loss Function

Loss function is a technique used to evaluate how well is the model (Janocha & Czarnecki, 2017). It works by computing the distance between the predicted output and the correct output. In this work, we used the categorical **cross-entropy** loss function. Categorical cross-entropy loss function is defined in equation (2.5).

$$-\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C 1_{y_i \in C_c} \log (p_{model}[y_i \in C_c]) \quad (0.3)$$

Where i iterates throughout N observations. c iterates throughout C classes. 1 is the indicator function that make one class as 1 and any other class as zero and iterates over C classes. $p_{model}[y_i \in C_c]$ predicts probability of observation i over c class. Categorical cross-entropy is used in this work because categorical cross-entropy is associated with the multiclass classification.

Regularization

A very important technique in neural network design is regularization. Regularization is used to prevent overfitting (Girosi, Jones, & Poggio, 1995). Overfitting is a problematic behavior in ML where the model predicts correctly in the training data but fail when tested with new data (Dietterich, 1995). Regularization works by summing the regularization penalty ($R(W)$) and the loss function ($\mathcal{L}(X, Y)$) as defined in equation (2.6).

$$L = \mathcal{L}(X, Y) + \lambda R(W) \quad (0.4)$$

The loss function could be weighted by the hyper-parameter (λ) to avoid the coefficients. The three most known regularization penalties L1, L2, and drop out. In this work dropout is used as a regularizer.

Gradient descent optimizers

One of the most popular optimizers in NN is gradient descent. The most popular gradient descent optimization techniques are stochastic gradient descent (SGD) (Ketkar, 2017), Adam (Kingma & Ba, 2014), Adadelta (Zeiler, 2012), SGD + momentum (Qian, 1999), Adagrad (Duchi, Hazan, & Singer, 2011) and Root Mean Squared Propagation (RMSprop) (Fakhry & Brery, 2022). These methods are used to find the local optimum of a differentiable function. The gradient descent is used to update the weights in each layer by using a backpropagation learning algorithm.

Adam is used in this work as the optimizer of the model. It is a combination of RMSP and gradient descent with a momentum optimizer. It is used in this work as it is an efficient optimizer because it does not require as much memory as other optimizers. The full algorithm of Adam is as shown in

```

Require:  $\alpha$ : Stepsize
Require:  $\beta_1, \beta_2 \in [0, 1)$ : Exponential decay rates for the moment estimates
Require:  $f(\theta)$ : Stochastic objective function with parameters  $\theta$ 
Require:  $\theta_0$ : Initial parameter vector
 $m_0 \leftarrow 0$  (Initialize 1st moment vector)
 $v_0 \leftarrow 0$  (Initialize 2nd moment vector)
 $t \leftarrow 0$  (Initialize timestep)
while  $\theta_t$  not converged do
   $t \leftarrow t + 1$ 
   $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  (Get gradients w.r.t. stochastic objective at timestep  $t$ )
   $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  (Update biased first moment estimate)
   $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update biased second raw moment estimate)
   $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)
   $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)
   $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameters)
end while
return  $\theta_t$  (Resulting parameters)

```

Fig.2.7.

Fig.0.1 Algorithm of computing Adam

In Adam algorithm, θ is the learnable parameters weights and biases. The hyperparameters in Adam algorithm includes α is the step size, the value from Momentum is represented as $\beta 1$, and $\beta 2$ represent RMSProp value. The algorithm starts by initializing m to 0 as the first moment vector. Then, v to 0 as the second moment vector. the last initializing is setting t to 0.

In every iteration (t), the algorithm starts by updating t to $t + 1$. Then, the gradients (g) with the respect to t is calculated. After that, the first moment m_t is updated in addition to the second moment v_t . Later, the bias-corrected m_t is computed as well as the bias-corrected v_t . Finally, the parameters θ get updated. The loop continues until Adam converge the parameters θ .

Backpropagation

Backpropagation is an algorithm where is backpropagates the errors from the output neurons to the input neurons (Hecht-Nielsen, 1992). It is used for training the weights in NN. Backpropagation is done in two steps. First propagation, it is the process of forward propagation in a training input in NN and generating the loss. Second weight update, it is done by multiplying the input activation function and the weights' delta to determine the gradient of the weight. Then the ratio of the gradient is subtracted from the weight.

الخلاصة

لغة الإشارة هي لغة التواصل الوحيدة المستعملة للتواصل بين الصم والبكم والمتحدثين الطبيعيين. تساعد لغة الإشارة الصم والبكم لغة الإشارة للتعبير عن مشاعرهم وافكارهم. تتعدد لغات الإشارة بتعدد اللغات المتحدثين بها كما ان كل لغة فيها لهجات مختلفة باختلاف المواقع الجغرافية. وسبب ذلك ضعف في التواصل بين الصم والبكم والاصحاء. هذه الرسالة تقدم حل مقترح لهذا الاشكال عن طريق صناعة تطبيق يتعرف ويترجم لغة الإشارة من الإشارات اليدوية الى حروف مكتوبة لحظياً. تم استعمال الشبكات العصبونية الالتفافية (Convolutional Neural Network) وتم استعمال مكتبة (OpenCV) لتطبيق الترجمة اللحظية. تم تدريب الشبكة باستعمال قاعدة البيانات (ArSL2018) والتي تحتوي على اكثر من ٥٠٠٠ صورة في ٣٢ صنف مختلف. حققت الشبكة المستعملة نسبة موزونة لكفاءة ودقة بنسبة ٩٧٪. كما ان متوسط المايكرو لكفاءة والدقة حققت نسبة ٩٧٪. وايضاً حققت الشبكة ٩٧٪ في الدقة. بعد ذلك تم استعمال OpenCV مع الشبكة المدربة لتفعيل خاصية الترجمة اللحظية. يبدأ البرنامج بالتقاط صورة من الكاميرا في الجهاز. بعد ذلك يتم ادخال الصورة في الشبكة المدربة للتصنيف وي نفس اللحظة يظهر البرنامج الحرف المتوقع على الشاشة. تم تطبيق البرنامج بشكل صحيح وبنتيجة مقبولة.

جامعة البحرين
كلية تقنية المعلومات



ترجمة فورية للغة الإشارة العربية إلى نص عربي باستخدام نموذج التعلم العميق

أطروحة مقدمة كجزء من متطلبات الحصول على درجة الماجستير
في لغة الآلة والحوسبة الذكية

إعداد

شيخه بنت إبراهيم بن محمد المانع
الرقم الجامعي: ٢٠٢١٠١٧٧٢

إشراف

أ. د. نبيل الحويحي
أستاذ
جامعة البحرين

المشرف المشارك

د. علاء الدين العمري
أستاذ مشارك
جامعة البحرين

مملكة البحرين

أبريل ٢٠٢٣ م