5-2023

The Automation of Translation Between Sign Language Variants Through Artificial Intelligence

Mohammad Sultan Buafra
bb5581@rit.edu

The Automation of Translation Between Sign Language Variants Through Artificial Intelligence

By

Mohammad Sultan Buafra

A Capstone Submitted in Partial Fulfilment of the Requirements for the

Degree of Master of Science in Professional Studies: Data Analytics

Rochester Institute of Technology

RIT Dubai

May 2023

RIT

Master of Science in Professional Studies:

Data Analytics

Graduate Capstone Approval

Student Name: **Mohammad Sultan Buafra**

Graduate Capstone Title: **The Automation of Translation Between Sign Language Variants Through Artificial Intelligence**

Graduate Capstone Committee:

Dr. Sanjay Modak	**Chair of Committee**	
Name	**Designation**	**Date**
Dr. Khalil AlHusseini	**Member of Committee/ Mentor**	
Name	**Designation**	**Date**

Acknowledgment

I would like to begin by emphasizing on my deep gratitude towards Dr. Khalil Al Husseini, my mentor, whose vision and empowerment pushed me towards achieving what I wouldn't have considered possible for myself. I would also like to thank all the faculty and fellow students that I have met and worked with during my studies. A special thanks to "The Outliers"; a group of brilliant students and friends whom I had the pleasure of working with during this period. Finally, I would like to thank my family, friends, and colleagues who gave me the strength, support, and space to continue my studies.

Abstract

Language is the method of communication between individuals and communities through either sounds, written symbols, or visual gestures. There is a variety of languages around the world. One significant language that is made up of hand gestures is sign language and it is the main method of communication between and with deaf people. Translation between languages is a practice that has been automated with the advancement of artificial intelligence, this has also allowed for the translation of sign language into written language. However, there remains a certain gap in translating between regional variants of sign language. The purpose of this study is to propose a method of implementing a machine learning model on multiple sets of data to automate the process of translating between regional variants of hand sign language, through a series of interconnected deep learning algorithms including Convolution Neural Networks & sequence Transformers. This study will produce a model that can translate between linguistic variants of hand sign language.

(Keywords: Sign Language, Translation, Neural Network, Convolution Neural Network, Transformer, Natural Language Processing, Transfer Learning)

Contents

Chapter 1: Introduction

1.1 Statement of Problem

The advancement of artificial intelligence and machine learning has allowed for the automated translation between over 100 textual and spoken languages, shattering the boundaries between individuals and communities through breaking the communication barrier and allowing cross-cultural/cross-lingual learning. This is not the case for sign language, as there is little to no automation between the regional variants of sign language.

Communication between sign language users of different variants remains difficult due to the similarity of sign gestures that represent different characters in different variants.

For example, Figure 1 shows a sign language gesture representing the letter "ك" in the Arabic sign language. The same gesture happens to also represent the letter "H" in the English sign language.

Figure 1: A hand gesture that represents two different letters depending on the language in use.

1.2 Background Information

Language, in the most basic form, can be defined as a system of sounds, written symbols, or visual gestures that make up the means of communication between individuals and within communities. There are over 7,000 spoken languages globally, 4,000 of which have written formats. One significant language that is made up of hand gestures is sign language.

Sign language is used as a means of communication between and with deaf people. While sign language is a language on its own, there are between 138 and 300 variants of sign language (Brooks 2018). The variance in sign language is due to the variance in written and spoken languages as well as socio geographical factors.

Translation between languages refers to transforming a sequence of words from one format into another without changing the meaning conveyed by the words being translated. This practice can be dated to over 1,000 years BC (Ulatus 2016). With the advancement of artificial intelligence, translation between languages has been conveniently automated, resulting in easier communications between individuals and communities belonging to different linguistic groups.

The advancement of machine learning has also allowed for the translation of sign language into written language. However, there remains a certain gap in translating between regional variants of sign language, causing difficulties in communication between deaf people belonging to different linguistic variants of the sign language.

1.3 Project Definition and Goals

The purpose of this study is to propose a method of implementing a machine learning model on multiple sets of data to automate the process of translating between regional variants of hand sign language.

The goals of the study are as follows:

1. To develop a model that can, to a certain degree of accuracy, translate between regional variants of hand sign language.
2. To have a measure of the effectiveness of the above-mentioned model.
3. To set a guideline for future research to enhance the above-mentioned model.

1.4 Project Impact

The impact of the model to be proposed in this study is two-fold, as follows:

1. Social Impact: The Automation of translation between regional variants of sign language will be one step closer to breaking communication barriers between sign language users of different regions. It will allow individuals to easily communicate with each other and understand each other, as is already done in spoken and textual languages over the phone.
2. Educational Impact: The automation of translation between regional variants of sign language will have a significant impact on the educational sector. Most linguistic educational tools rely on translation modules. Our model is a strong candidate as an educational tool that helps hearing and vocally impaired individuals communicate using new variants of sign language without relying on textual languages.

Chapter 2: Literature Review

Translating between languages is a field of study that varies in its concepts and theories which span over a long period dating back to 1,000 years BC. Over the course of time, different approaches to translation with different theories surrounding them. This variety of theories mainly focus one point; how far can the wording be deviated during the translation process to fit the meaning without changing the text as a whole (Ghanooni 2012). Translating between languages involves gathering information from one language and explaining the findings in another language and requires not only linguistic but cultural inferences (Birbili 2000)

Deep learning is a refined machine algorithm vastly used in speech recognition, natural language processing, and computer vision. It is also applied in designing intelligent machines or implementing machine learning. Applications of deep learning include generative, discriminative learning, and semi-supervised and active learning. It materializes in various forms, including artificial neural networks, Convolutional Neural Networks (CNN), and deep belief networks. Supervised learning is usually implemented using regression and classification techniques, while clustering algorithms implement unsupervised learning (Mishra and Gupta 2017).

With the advancement of artificial intelligence and machine learning, the concept of automating the process of translating between languages arose. The Georgetown-IBM experiment of translating between Russian & English in 1954 laid the foundation for machine translation and over the course of the next half century many machine translation techniques were improvised and implemented each with its own potentials and drawbacks (Nandasara, et al. 2019).

One of the major breakthroughs in the machine learning and the automation of the translation process is Natural Language Processing which is a field of machine learning which deals with language (Chowdhury 2003). Many natural language processing techniques are rooted within Recurrent Neural Networks, which are a subset of deep learning techniques, unlike normal neural networks and can handle a sequence of inputs. This is important in this field of learning due to the importance of having multiple inputs to model that emphasizes the relationship between such inputs (Singh and Mahmood 2021). Translation to images has been implemented using Neural networks in “Natural Language to Visualization by Neural Machine Translation” (Tang, et al. 2021)

Natural Machine Translation has evolved rapidly over the course of the few past years with the adoption or Recurrent Neural Networks to the adaptation of Convolution Neural Network models in the process (Chen, et al. 2018).

In “Attention Is All You Need” (Vaswani, et al. 2017) the idea of implementing a sequence Transformer model in Natural language Processing and Machine Translation is introduced. The model proved to be better in terms of accuracy and speed.

Convolution Neural Networks are neural networks that are used for image classification. The CNN architecture has several building blocks, including the convolution layer, pooling layer, fully connected layer, and the last layer activation function. Gradient descent and loss function are key algorithms used in training a network. The Convolution & Pooling layers allow the model to extract features of images used in the model while the fully connected layer is mainly used for classification purposes (Yamashita, et al. 2018).

One form of image classification that has been implemented and has had an impact is the translation of sign language into text. This has aided communication between the hearing impaired and the general population. In “A Review on Arabic Sign Language Translator Systems” (Mohammed and Kadhem 2021) different approaches to sign language recognition and translation are discussed based on capturing methods for Arabic Alphabets in sign language and in “Sign Language Recognition Using Convolutional Neural Networks” (Pigou, et al. 2015) a method of utilizing a Convolution Neural Network in sign language translation to text is proposed and implemented. In “Adversarial text-to-image synthesis: A review” (Frolov, et al. 2021) the usage of more complex networks and methodologies have been implemented to procure images through text. However, the process and results faced many challenges.

In A “Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects” (Zewen , et al. 2020) the layers of a Convolution Neural Network are broken down into three basic essential layers. Convolution layers are the first layers in the network and are tasked with feature extraction from images. Pooling layers are included to reduce computations and allow the model to be faster and more accurate and Fully connected layers, the most basic layers of neural networks are tasked with the actual classification function.

The usage of optimizers in a Convolution Neural network is discussed in “Optimization and acceleration of convolutional neural networks: A survey” (Habib and Qureshi 2022) and are used to enhance computation and speed of the network. Adam optimizer is introduced in “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION” (Kingma and Ba 2015) as a stochastic gradient descent optimizer; A feature that automates the process of adapting the weights in a neural network to reduce loss. In “Comparative Study of Optimizers in the Training of a Convolutional Neural Network in a Binary Recognition Mode” (Lopez-Sanchez, et al. 2022) three optimizers were compared in the use of in a Convolution Neural Network for image recognition with Adam inducing the best performance while training the network.

Activation functions are discussed in “ACTIVATION FUNCTIONS IN NEURAL NETWORKS” (Sharma, Sharma and Athaiya 2022) and are defined as Gatekeepers that specially used in artificial neural networks to transform an input signal into an output signal which in turn is fed as input to the next layer in the stack. The various types of activation functions introduced over time are discussed as well. In “Deep Learning using Rectified Linear Units (Agarap n.d.) Rectified Linear Units (Relu) activation function is introduced and discussed and in “Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU” (Zhang, et al. 2018) the Relu activation function is used in a Convolution Neural network that predicts multiple sclerosis using brain scans.

Transfer learning is defined as the transfer of pretrained models trained on readily available data to the use on different cases within the same scope (Weiss, Khoshgoftaar and Wang 2016). “Transfer Learning for Low-Resource Neural Machine Translation” (Zoph, et al. 2016) shows that using a pretrained model for Neural Machine Translation improves the results of the study while reducing the complexity and resources required for the study and in “Evolution of transfer learning in natural language processing” (Malte and Ratadiya 2019) the history and different methodologies used in implementing transfer learning in natural language processing models are dissected.

The main takeaways of this literature review are as follows:

1. Translation between languages is not only a practice but a field of study with many theories and approaches around it.
2. With the advancement of Artificial Intelligence and Machine learning arose the implementation of Machine Translation, the automation of language translation under the subset of Natural Language Processing
3. Natural Language Processing which is a field of machine learning which deals with language, speech, and text.
   Most recent Advancements in Machine Translation include the adaptation of Sequence Transformers in translation.
4. Image classification has been used in the process of translating Sign Language into text, and the most notable use is in Convolution Neural Networks which have layers that extract features from images to best the classification process.
5. Optimizers are functions used in training neural networks that adjust the weights and biases within a neural network to create more accurate results.
6. There are different types of activation functions that can be imbedded in the layers of neural networks with the aim of shutting and opening certain neurons or paths to better the network.
7. Transfer learning is a method of utilizing pre trained models on new cases with the aim of reducing time spent on training new networks and accessing more reliable and useable networks.

Chapter 3: Project Description

3.1 Methodology

The purpose of this study is to propose a machine learning-based model that automates the process of translation between regional variants of hand sign language through a series of interconnected deep learning algorithms. The overall method is depicted in Figure 2.

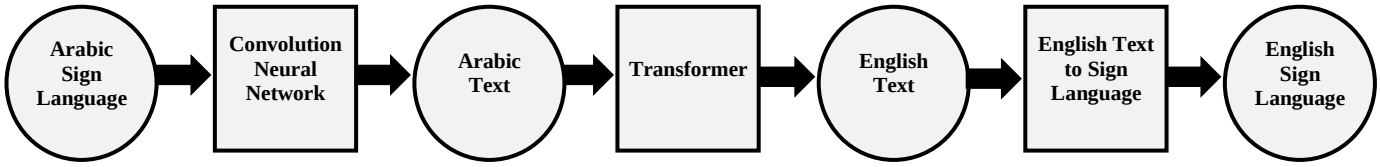

Figure 1: Overview of the proposed method

As a starting point, and to tackle definitive tasks that achieve a minimally viable product, we currently focus on Arabic to English sign languages translation.

Our proposed model takes as input a sequence of Arabic sign language images that make up a word and, through the interconnected deep layers, output the translated sequence of English sign language images. To summarize:

1. The first layer consists of a Convolution Neural Network that is fed a dataset of Arabic sign language alphabet images. The Convolution Neural Network extracts the features of Arabic sign language alphabet images in the training portion of the dataset and is used to classify the images in both training and testing portions of the datasets into their corresponding Arabic alphabet text. The Convolution Neural Network Outputs Arabic Text.
2. The following layer consists of transforming the outputted Arabic text into English text sequence through a Transformer for Natural Language Processing.
3. The final layer utilizes the English text outputted from the Transformer as an input for a program that transforms the English text into English sign language images.

3.2 Dataset Analysis

For the purpose of this study, two publicly available dataset will be utilized:

1. “zArASL_Database_54K”: a dataset consisting of 54,049 images of the 32 Arabic alphabet sign language. The images captured are of 40 different individuals across different age groups. This dataset is available on Kaggle Database. Table 1 shows a brief summary the dataset while figure 3 shows a sample taken from the dataset.

Class ID	Arabic Alphabet	Instances
0	ع	2,114
1	ال	1,343
2	أ	1,672
3	ب	1,791
4	د	1,634
5	ظ	1,723
6	ض	1,670
7	ف	1,955
8	ق	1,705
9	غ	1,977
10	ه	1,592
11	ح	1,526
12	ج	1,552
13	ك	1,774
14	خ	1,607
15	لا	1,746
16	ل	1,832
17	م	1,765
18	ن	1,819
19	ر	1,659
20	ص	1,895
21	س	1,638
22	ش	1,507
23	ط	1,816
24	ت	1,838
25	ث	1,766
26	ذ	1,582
27	ة	1,791
28	و	1,371
29	ى	1,722
30	ي	1,293
31	ز	1,374
Total		**54,049**

Table 1: Summary of zArASL_Database_54K dataset.

Figure 3: A sample image of a hand gesture that represents the letter "ب" in the zArASL_Database_54K dataset.

2. “American Sign Language Dataset” is a dataset consisting of 34,627 images of the English alphabet sign language. This dataset is available on Kaggle Database. Table 2 shows a brief summary the dataset while figure 4 shows a sample taken from the dataset.

Class ID	English Alphabet	Instances
0	0	70
1	1	70
2	2	70
3	3	70
4	4	70
5	5	70
6	6	70
7	7	70
8	8	70
9	9	70
10	A	70
11	B	70
12	C	70
13	D	70
14	E	70
15	F	70
16	G	70
17	H	70
18	I	70
19	J	70
20	K	70
21	L	70
22	M	70
23	N	70
24	O	70
25	P	70
26	Q	70
27	R	70
28	S	70
29	T	70
30	U	70
31	V	70
32	W	70
33	X	70
34	Y	70
35	Z	70
Total		**2,240**

Table 2: Summary of American Sign Language Dataset.

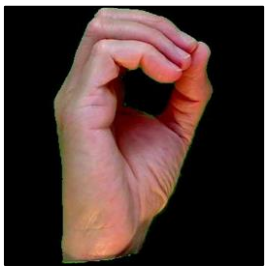

Figure 4: A sample image of a hand gesture that represents the letter "O" in the American Sign Language Dataset.

3.3 Data Pre-Processing

The first step in the data pre-processing stage is to create a directory in the main Python environment containing a subset of the "American Sign Language Dataset" to be used in the study. The subset taken contains one image of each American sign language alphabet letter named as the letter (E.g.: A.jpg). Numbers are excluded from the directory as this study is focused on translating letters. The rationale behind creating this directory is to ease the process of transforming the English Alphabet into its corresponding American sign language image at a later stage in the model.

The next step in the data pre-processing stage is to download the "zArASL_Database_54K" into the Python environment. This is done importing the opendatasets library and using the od.download() function to directly download the dataset from Kaggle into the environment. Once the dataset is downloaded, we can view a summary of the dataset as shown in figure 5 by importing the Pandas library and using the pd.read() function.

	id	ClassId	ImagePath
0	1	0	ArASL_Database_54K/ain/AIN (1).JPG
1	2	0	ArASL_Database_54K/ain/AIN (10).JPG
2	3	0	ArASL_Database_54K/ain/AIN (100).jpg
3	4	0	ArASL_Database_54K/ain/AIN (1000).JPG
4	5	0	ArASL_Database_54K/ain/AIN (1001).JPG

Figure 5: A summary of the "zArASL_Database_54K" in the Python environment

The following step is to create the data frame to be used in training and testing the Convolution Neural Network. It is important to note that during this step, we replace the numerical Class ID with Arabic Alphabet class ID through creating a list of the Arabic alphabet in the environment and passing it through dataset. We also remove other variables keeping only the newly updated Class ID and the image path as shown in figure 6.

	img	class
0	ArASL_Database_54K/ain/AIN (1).JPG	ع
1	ArASL_Database_54K/ain/AIN (10).JPG	ع
2	ArASL_Database_54K/ain/AIN (100).jpg	ع
3	ArASL_Database_54K/ain/AIN (1000).JPG	ع
4	ArASL_Database_54K/ain/AIN (1001).JPG	ع

Figure 6: A summary of the data frame created.

From the Sklearn Library, the train_test_split function is used to split the data frame into training data and testing data with a margin of 0.1 for the test size as the dataset is relatively large. Shuffle is set to be true to increase the model's accuracy and reduce bias through shuffling the datasets in each epoch the model is training on. This is important because it allows for the model to take in the data in a random and not systematic order. The data frame is split into 48,644 images for the training data and 5,405 images for the testing data as shown in figure 7.

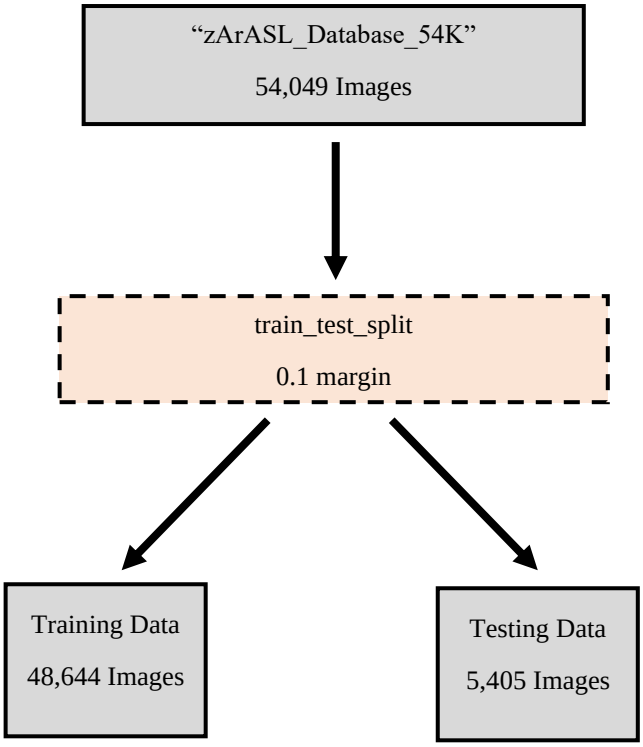

Figure 7: A representation of the train_test_split function splitting the "zArASL_Database_54K" into training and testing data.

3.4 Convolution Neural Network

The first step in creating in creating the convolution neural network is setting the hyperparameters of the network. Hyperparameters are variables that set the structure of network and define how the network is trained. The below hyperparameters are set for this convolution neural network:

- Size defines the size of the image through the training of the network and is set to 64 as this is the original size of the images.
- Channel refers to the depth of the image to be trained on. Since this dataset contains greyscale images the channel is set to one as opposed to training images with colour on three channels.
- Batch refers to the number of images entering the network for training at a time. Note that a batch of 128 is set for this neural network is rather large as opposed to standard batch sizes of 32 or 64.

- Epochs refers to the number of times the neural network trains on the entire dataset. A large number for epochs means that the neural network will train for a longer time but helps reach better results. This network was trained for 25 epochs.

The next step in creating the convolution neural network is to set the training and testing data. This is done in order to normalize the data so that is exactly similar in terms of dimensions and size before entering the network. Note that certain networks allow for augmentation of the images through flip, rotation, and recoloring, to allow the network to train more accurately on different types of images. However, since this project relies on hand sign gestures which might have different inferences if flipped or rotated, no augmentation was added.

The training and testing dataset were created using the Keras ImageDataGenerator with the following parameters.

- Directory: The directory created for the images to be used in the neural network
- The X and Y axis's representation of the image and its class
- The size of the image to be entered which was defined earlier in hyperparameters.
- Colour Mode: the colour of the image corresponding to the channels the image has.
- Shuffle: Set to true to shuffle the images in each batch entering the network for better accuracy.
- Batch Size: The size of the batch of images entering the network at a time which is set in the hyperparameters earlier.

The following step is to import the required libraries to set the structure of the Convolution neural network and create the model itself. A Convolution neural network is normally structured on multiple instances of three basic layers:

- Convolution Layer: The most significant layer in a convolution neural network and has the function of extracting features from images to be able to classify the images into classes based on similarities in the features extracted. For this study 4 basic 2D convolution layer is implemented.
- Pooling Layer: is the layer used to reduce the output of the convolution layer easing and speeding the computation. 4 Max 2D pooling are implemented in this study which takes the maximum number in each kernel outputted by the convolution layer to downsize.
- Fully Connected Layer: Are the basic layers of neural networks with the function of computing the output of the layers above and classifying them according to the desired output. 4 Fully connected layers are included in this model's structure with a final output of 32 units corresponding to the 32 classes of the Arabic Alphabet

In addition to the below layers, it's important to note the following important features in a convolution neural network:

- Optimizer: A feature that automates the process of adapting the weights in a neural network to reduce loss. For this study an Adam optimizer with a default learning rate of 0.001 is set
- Activation function: is a function inserted into a neural network that decides on closing and opening certain neurons to ease and better the computation while training the neural network. The most widely used Relu linear activation function is added to the layers of this convolution neural network.

The below figure 8 depicts the structure of the Convolution Neural Network

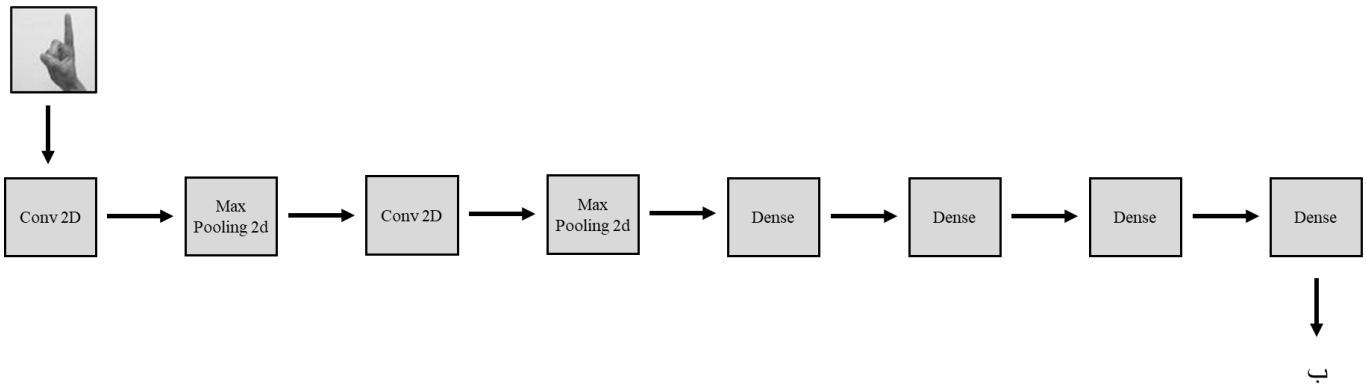

Figure 8: The structure of the Convolution Neural Network created.

The model is then trained for 25 epochs, reaching the best result on its 16th epoch as shown in the below table 3.

Epoch	Time (In Minutes)	Loss	Accuracy	Validation Loss	Validation Accuracy
1	38.58	0.3883	0.8813	0.0912	0.978
2	38.62	0.3737	0.8854	0.0933	0.9784
3	38.93	0.3654	0.8884	0.0708	0.98
4	38.58	0.3487	0.8924	0.1072	0.9761
5	38.62	0.3428	0.8957	0.0843	0.9789
6	38.98	0.3362	0.896	0.0662	0.98
7	38.75	0.3263	0.8999	0.0841	0.9785
8	38.57	0.3181	0.9031	0.0693	0.9806
9	38.73	0.3186	0.9038	0.0665	0.9795
10	38.98	0.3142	0.9044	0.0716	0.9796
11	39.07	0.3099	0.9067	0.0668	0.9808
12	39.65	0.3023	0.9064	0.0666	0.9798
13	39.43	0.2992	0.9089	0.0918	0.9806
14	39.70	0.2952	0.9104	0.0603	0.9811
15	40.67	0.2939	0.9106	0.0724	0.9798
16	41.37	0.2902	0.9123	0.0622	0.9809
17	39.62	0.2978	0.9101	0.0626	0.9809
18	39.17	0.2905	0.911	0.0762	0.9798
19	39.45	0.2912	0.911	0.0638	0.9802
20	39.27	0.2897	0.911	0.0652	0.9795
21	39.47	0.2946	0.9095	0.0625	0.9802
22	39.37	0.2922	0.9108	0.0618	0.9808
23	39.30	0.2889	0.9112	0.0612	0.9804
24	39.63	0.2973	0.9097	0.0699	0.9806
25	39.65	0.3002	0.9088	0.0692	0.9802

Table 3: The results of the training of the Convolution Neural Network.

The model is then saved for usage in the study using the model.save function as APTET.H5.

The final step is to use the model in the actual study through loading it in the main environment using the models.load_model function and using APTET.predict on a certain image by directing it to this image's path to predict the images Arabic alphabet.

3.5 Transformer

The second layer in the model is a Transformer that uses the output of the Convolution Neural Network above as an input and outputs the corresponding English alphabet letter. Transfer learning refers to the method of using a pretrained model used in similar cases to solve a new problem. This is done by utilizing the model with through introduction to new testing data and fine tuning the parameters to receive desirable results For this study, transfer learning is chosen as opposed to the traditional method of training and deploying a Transformer or simply routing the Arabic alphabet to the English alphabet. The reason for doing so lies in that training a Transformer for natural language processing can be time consuming, complex and requires tremendous computing power. Routing using simple coding would be easier in this case, however as this model is intended to be the building block for future research with more complex translations such as complete words, a Transformer was chosen. In this study, the pretrained model used is the Helsinki Natural Language Processing model. As the Helsinki Natural Language Processing model is a model trained and deployed on translation Arabic language into English language with an accuracy rate of 0.97. The below table 4 summarizes the Helsinki Natural Language Processing model.

Model Name	Helsinki-NLP/opus-mt-ar-en
Model Type	Sequence Transformer
Training Dataset	Opus
Training Accuracy	0.97
Input	Arabic Language text
Output	English Language text
Pre-processing methods	Tokenization

Table 4: A summary of the Helsinki NLP model used in the study.

There are a variety of means for adding a pretrained model to a Python environment. In this study the model is added to the main environment as the second layer model_checkpoint() and pipeline() functions with the input set to be the output of the Convolution Neural Network. The output of the Transformer is as depicted in figure 15.

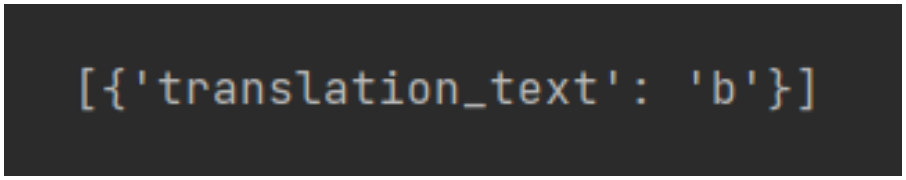

Figure 9: A sample of the output of the transformer.

Note that the output above is a list. To be able to use this in the model, there are multiple steps that need to be taken. First the list is transformed into a string by using the str() function on the result. The following step is to split the newly created string using the slice() function to get the letter on its own as depicted in the below figure 16.

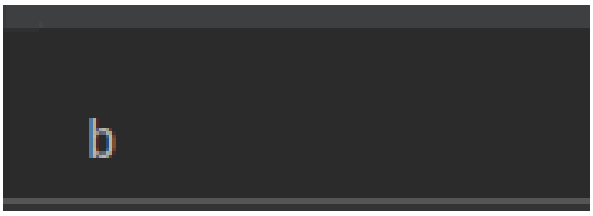

Figure 10: The required result after using the str() and slice() functions.

3.6 Results

The final layer of the model consists of transforming the outputted English alphabet letter into its corresponding English Sign Language image and displaying said image. The implementation of this layer is quite simple and requires only simple coding. Procuring the result is done through recalling the image corresponding to the letter outputted earlier and using MatplotLib imgread() function routed to the directory set in the data pre-processing stage. The result is depicted in the below figure 17.

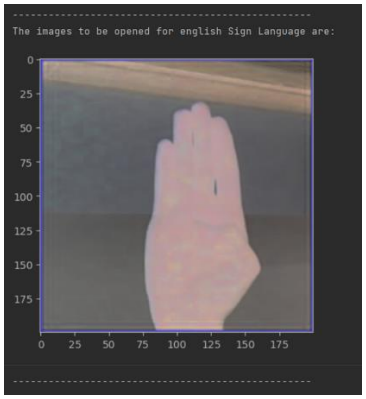

Figure 11: The final output of the model; the translated English Sign Language Alphabet.

3.7 Analysis of Results

The resulting model is able to translate a single image of an Arabic hand sign language alphabet into a single English hand sign language alphabet through three different layers and as depicted in the below figure 12.

Figure 12: The process of flow of the model from input to output.

The full model was tested multiple times showing the results in the below table 5.

Test	Input	Target Arabic Alphabet	Actual Arabic Alphabet	Target English Alphabet	Actual English Alphabet	Target Output	Actual Output
1		ب	ب	B	B		
2		أ	أ	A	A		
3		د	خ	D	K		
4		ز	ز	Z	Z		

Table 5: The target and actual results for multiple tests on the full model.

From the above, we can infer that the model passed 75% of the testing phase, and a closer look on the test the model failed showed that the model's convolution neural network incorrectly classified the letter. Looking at the image for both letters in the below figure 13 shows that there is some similarity between the letters which might mean that the Convolution Neural network requires some adjustment in order to better infer and classify.

Figure 12: Right Side: The letter خ In Arabic Sign Language while Left Side: The letter د In Arabic Sign Language

Chapter 4: Conclusion

4.1 Conclusion

In conclusion, while there is a gap in translating between regional variants of sign language, the model proposed and implemented lays a setting stone in filling this gap through proving effective in being able to translate Arabic Sign Language Alphabet into English Sign language alphabet through a three-layered model that includes a Convolution Neural Network, a Transformer, and simple coding. While this primitive model is able to translate letter to letter only, it lays the foundation for future research to further the cause.

4.2 Limitations

The model has the below limitations:

1. The full model needs to be manually operated. The separate layers in the models are not connected. Connecting the layers into a single model requires complex coding.
2. The model is currently trained into translating a single image at a time and cannot compute full words or sentences. Doing so requires advanced techniques and computation power not available at the moment.
3. For the model to be fully functional, it requires Application Programming Interface for deployment to be used by end users.

4.3 Future work

Future work on furthering this study is already in process as this study has been chosen by the Education Technology Accelerator to be funded to be studied in coordination with researches being studied by The American Sign Language for Second Language Learners Research Lab within the National Technical Institute for the Deaf's research. Possible future enhancements the research may take include but are not limited to.

1. Enhancement of the current model's accuracy through restructuring the layers of the model through building a more complex Convolution Neural Network, replacing the pre-trained transformer with an in-house model, etc…
2. Introducing enhancements to the model in terms of the ability to translate complete words and sentences, translating between languages and the inclusion of more languages, translating from live feeds of videos, etc…

References

Birbili, Maria. 2000. "Translating from one language to another."

Brooks, Richard. 2018. "A Guide to the Different Types of Sign Language Around the World." *The Language Blog*, 10 May: 1.

Chen, Mia Xu, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Machery, George Foster, Llion Jones, et al. 2018. "The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation."

Chowdhury, Gobinda G. 2003. "Natural Language Processing ."

Frolov, Stanislav, Tobias Hinz, Federico Raue, Jorn Hees , and Andreas Dengal. 2021. "Adversarial text-to-image synthesis: A review."

Ghanooni, Ali Reza. 2012. "A Review of the History of Translation Studies." *Theory and Practice in Language Stdies, Vol. 2* 8.

Habib, Gousia, and Shaima Qureshi. 2022. "Optimization and acceleration of convolutional neural networks: A survey." *Journal of King Saud University - Computer and Information Sciences* 24.

Kaplan, Dylan. 2022. "Keras Shuffle: A full in-depth guide." 1 June.

Kingma, Diedrik P., and Jimmy Lei Ba. 2015. "ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION."

Lopez-Sanchez, Marco, Jose Hernandez-Torruco, Betania Hernandez-Ocana, and Oscar Chavez-Bosquez. 2022. "Comparative Study of Optimizers in the Training of a Convolutional Neural Network in Binary Recognition Model."

Malte, Aditya, and Pratik Ratadiya. 2019. "Evolution of transfer learning in natural language processing."

Mishra, Chandrahas, and D. L. Gupta. 2017. "Deep Machine Learning and Neural Networks: An Overview." *IAES International Journal of Artificial Intelligence (IJ-AI).*

Mohammed, R. M., and S. M. Kadhem. 2021. *A Review on Arabic Sign Language Translator Systems.* Iraqi Academics Syndicate International Conference for Pure and Applied Sciences.

Nandasara, S. T., Yoshiki Mikami, AIC. Mohideen, and K. G. D. Tharangie. 2019. "Automated Language Translation: Opportunities and." *International Journal of Computer Applications* 7.

Pigou, Lionel, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. 2015. "Sign Language Recognition Using Convolutional Neural Networks."

Sharma, Siddharth, Simone Sharma, and Anidhya Athaiya. 2022. "International Journal of Engineering Applied Sciences and Technology, 2020." *International Journal of Engineering Applied Sciences and Technology.*

Singh, Sushant, and Ausif Mahmood. 2021. *The NLP Cookbook: Modern Recipes for.* IEEE.

Tang, Nan, Yuyu Luo, Guoliang Li, Chengliang Chai, and Xuedi Qin. 2021. "Natural Language to Visualization by Neural Machine Translation."

Ulatus. 2016. "The Evolution of Translation." *Machine Translation, Translation, Translation Culture*, 16 April: 1.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomze, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need."

Weiss, Karl, Taghi M. Khoshgoftaar, and DingDing Wang. 2016. "A survey of transfer learning."

Yamashita, Rikiya, Mizuho Nishio, Ricahrd Kinh Gian Do, and Kaori Togashi. 2018. "Convolutional neural networks: an overview."

Zewen , Li, Yang Wenjie, Peng Shouheng, and Liu Fan . 2020. "A Survey of Convolutional Neural Networks:."

Zhang, Yu-Dong, Chichun Pan, Junding Sun, and Chaosheng Tang. 2018. "Multiple sclerosis identification by convolutional neural network with dropout and parametric ReLU." *Journal of Computational Science* 10.

Zoph, Barret, Deniz Yuret, Jonathan May, and Kevin Knight. 2016. "Transfer Learning for Low-Resource Neural Machine Translation."

Appendix

Code

Data Pre-Processing

```
pip install kaggle

#%%

pip install opendatasets

#%%

import opendatasets as od

#%%

od.download('https://www.kaggle.com/datasets/zssash/zarasl-database-54k')

#%%

arabic_alphabet = ['ع', 'ال', 'أ', 'ب', 'د', 'ظ', 'ض', 'ف', 'ق', 'غ', 'ه', 'ح', 'ج',
'ك', 'خ', 'لا', 'ل', 'م', 'ن', 'ر', 'ص', 'س', 'ش', 'ط', 'ت', 'ث', 'ذ', 'ة', 'و', 'ئ',
'ي', 'ز']

#%%

train_path = ".\zarasl-database-54k"

#%%

import pandas as pd

#%%

data = pd.read_csv("zarasl-database-54k/Labels/ImagesClassPath.csv")

data.head()

#%%

for i in range(data.shape[0]):

    data.loc[i, 'ClassId'] = arabic_alphabet[data.loc[i, 'ClassId']]

data.head(-5)

#%%

df = pd.DataFrame({'img':data['ImagePath'], 'class':data['ClassId']})

df.head()

#%%

from sklearn.model_selection import train_test_split

#%%

df_train, df_test = train_test_split(df, test_size=0.1, shuffle=True)
```

```
print('Train shape:', df_train.shape)

print('Test shape:', df_test.shape)
```

Convolution Neural Network

```
size  = 64

channels = 1

batch = 128

epochs = 25

steps_per_epoch=  data.shape[0] // batch

#%%

pip install tensorflow --upgrade

#%%

pip install keras --upgrade

#%%

X_train = train_datagen.flow_from_dataframe(df_train, directory = train_path, x_col =
'img',y_col = 'class',target_size = (size,size),color_mode = 'grayscale', shuffle =
True, batch_size = batch)

X_test = test_datagen.flow_from_dataframe(df_test, directory = train_path, x_col =
'img', y_col = 'class', target_size= (size,size), color_mode = 'grayscale', shuffle =
True, batch_size = batch)

#%%

from tensorflow.keras.optimizers import Adam

#%%

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau

#%%

optimizer = Adam(learning_rate=0.001)



callback_earlystopping = EarlyStopping(monitor='val_accuracy', mode='max',
min_delta=0.003, patience=25)

callback_learningrate = ReduceLROnPlateau(monitor='accuracy', mode='max',
min_delta=0.01, patience=5, factor=.05, verbose=1)



callbacks = [callback_earlystopping, callback_learningrate]
```

```
#%%

X_train.class_indices

#%%

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten,
Dense, Dropout

from tensorflow.keras.models import load_model

#%%

Model = Sequential([

    Conv2D(filters=32,  kernel_size=(3,3), activation="relu",
input_shape=(size,size,channels)),

    BatchNormalization(),

    MaxPool2D(2,2, padding='same'),

    Dropout(0.25),


    Conv2D(filters=128,  kernel_size=(3,3), activation="relu"),

    BatchNormalization(),

    MaxPool2D(2,2, padding='same'),

    Dropout(0.25),


    Conv2D(filters=512,  kernel_size=(3,3), activation="relu"),

    BatchNormalization(),

    MaxPool2D(2,2, padding='same'),

    Dropout(0.25),


    Conv2D(filters=2048,  kernel_size=(3,3), activation="relu"),

    BatchNormalization(),

    MaxPool2D(2,2, padding='same'),

    Dropout(0.25),


    Flatten(),

    BatchNormalization(),
```

```
    Dense(units=4096, activation="relu"),

    BatchNormalization(),

    Dropout(0.25),



    Dense(units=1024, activation="relu"),

    BatchNormalization(),

    Dropout(0.25),



    Dense(units=256, activation="relu"),

    BatchNormalization(),

    Dropout(0.25),



    Dense(units=64, activation="relu"),

    BatchNormalization(),

    Dropout(0.5),



    Dense(units=32, activation="softmax"),

])




Model.compile(optimizer=optimizer, loss="categorical_crossentropy",
metrics=["accuracy"])

Model.summary()

#%%

history = Model.fit(X_train, validation_data=X_test,
epochs=epochs,  callbacks=callbacks)

#%%

Model.save("APTET.h5")
arabic_alphabet = ['ع', 'ال', 'أ', 'ب', 'د', 'ظ', 'ض', 'ف', 'ق', 'غ', 'ه', 'ح', 'ج',
'ك', 'خ', 'لا', 'ل', 'م', 'ن', 'ر', 'ص', 'س', 'ش', 'ط', 'ت', 'ث', 'ذ', 'ة', 'و', 'ئ',
'ي', 'ز']

#%%

img = Image.open(images_path)

#%%
```

```
img_arr = np.array(img).reshape(1,64,64,1)

#%%

APTET=models.load_model("APTET.h5")

#%%

 out = APTET.predict(img_arr).argmax()

char = arabic_alphabet[out]

print (char)
```

Transformer

```
from transformers import pipeline
model_checkpoint = "Helsinki-NLP/opus-mt-ar-en"

translator = pipeline("translation", model=model_checkpoint)

EnglishLetter = translator(char)

print (EnglishLetter)
```

Results

```
print(EnglishLetter)

#%%

XY=str(EnglishLetter)

print (XY)

#%%

S2=slice(23,24)

#%%

L=(XY[S2])

print(L)

#%%

print("-"*50)

print("The images to be opened for english Sign Language are:")

image = mpimg.imread('EnglishAlphabets/[L].jpg')

plt.imshow(image)

plt.show()

print("-"*50)
```