Tri-Lingual Sign Language Translator

Shivangi Sharma
sharshiv
sharshiv@iu.edu
Indiana University,
Bloomington

Sravya Vujjini
svujjin
svujjin@iu.edu
Indiana University,
Bloomington

Drumil Joshi
dtjoshi
dtjoshi@iu.edu
Indiana University,
Bloomington

Abstract

Most hearing-impaired and mute people use sign language to communicate both inside and outside of their social groupings. Recognition of Sign Language(SLR)[1], is intended to identify acquired hand motions and to continue until related hand gestures are translated. The fact that some or all of the ordinary people do not speak this language and that every country has its sign language creates a barrier to communication. We have created a model that can understand the gestures and translate them to cater to a wider audience. By developing Deep Neural Network[2] architectures, the model will learn to detect alphabets and once the model successfully recognizes[3] the gesture, a relevant letter is generated. This paper will be a two-fold endeavor with comparative study as the research part and Tri-Lingual sign language translator, as the application part.

1 Introduction

An effective communication tool is sign language for the population that has difficulty communicating due to hearing, or speaking challenges. Hand gestures are used in sign language to communicate ideas and thoughts visually. There are between 138 and 300 different sign languages being used all over the world. Over 5% of the global populace consists of people with partial or fully disabling hearing or speaking loss. However, due to the profession of a sign language interpreter being the 55th most widely and onerously licensed one, it creates a scarcity of experts which further makes it difficult to teach and learn sign language. Communication in sign language is complex and covers hand gestures, body language, and facial expressions. In this paper, we will be concentrating on hand gestures based on ASL[4], GSL[5], and ISL[6].

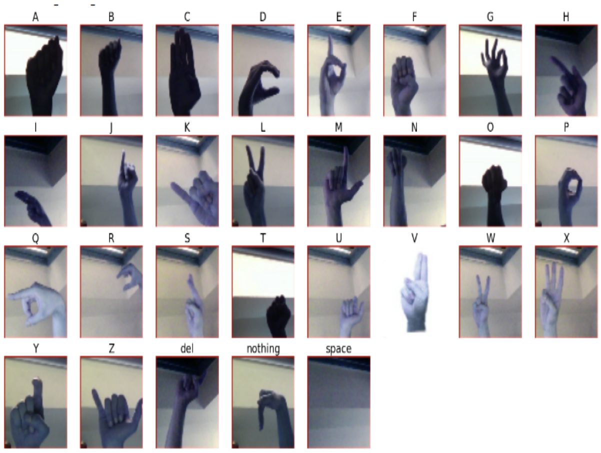


Figure 1: Sign Language Hand Gestures

1.1 Motivation

The isolation that the impaired population experiences are what spurs this project's ambition. The impaired population has higher rates of loneliness and depression[7], particularly significant hurdles that negatively impact life quality are caused by the communication gap between the impaired population and the rest of the population. This paper can also aid systems that translate[8] hand gestures into text that are commonly utilized in public places including hospitals, airports etc.

1.2 Problem Statement

Sign language has particular motions for each letter of the English alphabet and based on them, two categories of sign languages exist: Static Gesture[9] and Dynamic Gesture[10]. The dynamic gesture is utilized for specific concepts whereas, the static gesture is used to symbolize the alphabet and numbers. Additionally, dynamic gesture encompasses phrases, clauses, etc. The difference between static and dynamic gestures lies in the movements of the hand, the head, or both. Despite extensive research efforts over the past few decades, designing a sign language translator remains a difficult task and when done for multiple languages, the difficulty level increases extensively. Additionally, even identical signs can appear very differently depending on the things like angle from which they are viewed, the signer, etc. The study's main objective is to employ deep learning, machine learning, and transfer learning to identify the best performing model based on the defined metrics and utilize that to create a static sign language translator. This research will become a stepping stone for the future study of a dynamic tri-lingual sign language translator[11].

1.3 Objectives

Our approach focuses on promoting automatic sign language translation to overcome communication and learning barriers. This research aims to recognize hand gestures that include 26 English alphabet letters(A-Z).

2 Implementation

2.1 Dataset

We have integrated the American, German, and Indian Sign Languages hand gestures. The training dataset comprises of 1,15,000 photos of 29 different classes. 26 of the classes correspond to the letters A through Z and three classes for SPACE, DELETE, and NOTHING. For our application purposes, SPACE, NOTHING, and DELETE were not used and only 26 classes for letters were defined.

2.2 Data Preprocessing

The procedures we used for image preprocessing[12] are as follows:
1. Read images
2. Make all of the photos the same size/form
3. Eliminate noise
4. By dividing the picture array by 255, all image pixel arrays are transformed to values between 0 and 255
A sample of the same can be seen in Figure 2.

2.3 Artificial Neural Networks

Neural networks[13], which are a key functional element of deep learning, are renowned for modeling human brain functioning to tackle complex data-driven problems. Multiple layers of artificial neurons are fully connnected that process the input data to produce the desired output.

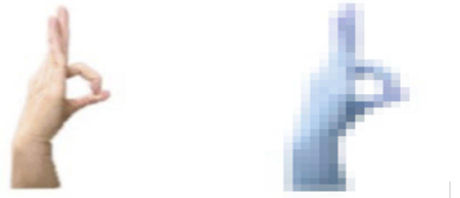


Figure 2: Sample Image before and after Preprocessing

The following are the artificial neural network's primary elements:
Input Layer: It introduces the initial data into the system for later processing by other layers of artificial neurons.[14].
Hidden Layer: Between the input and output layers of an artificial neural network is a layer known as the hidden layer, wherein artificial neurons process a collection of weighted inputs to generate an output using an activation function.
Output Layer: The output layer makes a final prediction using information from prior hidden layers.

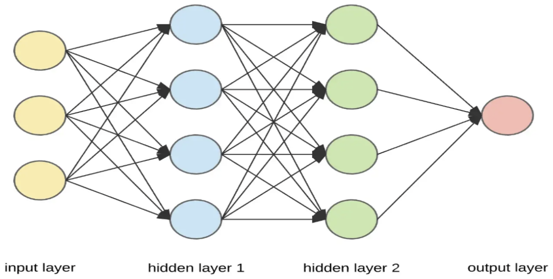


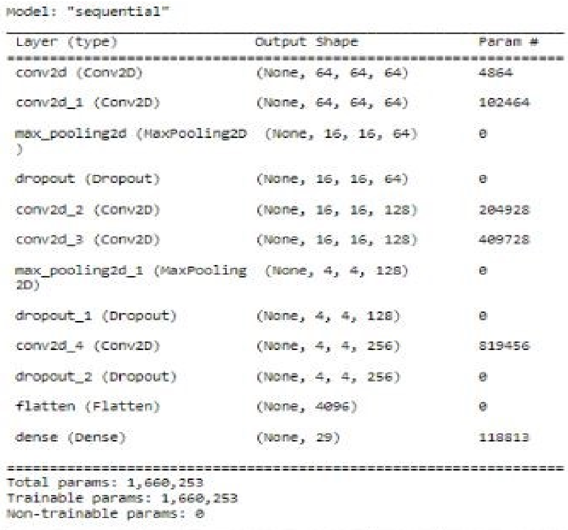
Figure 3: Artificial Neural Network

2.4 Convolutional Neural Network (CNN)

A Convolutional Neural Network (CNN / ConvNet)[15] is a Deep Learning model that takes the image as an input, allocate the different components of the image some learnable weights and biases, and provides the desired output after classification.
Below is the description of the layers used in the Convolutional Neural Network:

2.4.1 Convolutional Layer

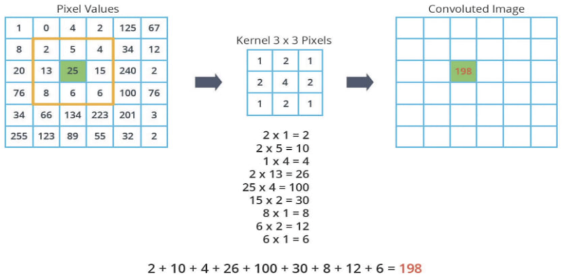
It is the center structure block of CNN[13] which becomes more complicated in each layer, distin-



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 64, 64, 64)        4864

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        102464

 max_pooling2d (MaxPooling2D  (None, 16, 16, 64)       0
 )

 dropout (Dropout)           (None, 16, 16, 64)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 128)       204928

 conv2d_3 (Conv2D)           (None, 16, 16, 128)       409728

 max_pooling2d_1 (MaxPooling  (None, 4, 4, 128)        0
 2D)

 dropout_1 (Dropout)         (None, 4, 4, 128)         0

 conv2d_4 (Conv2D)           (None, 4, 4, 256)         819456

 dropout_2 (Dropout)         (None, 4, 4, 256)         0

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 29)                118813

=================================================================
Total params: 1,660,253
Trainable params: 1,660,253
Non-trainable params: 0
```

Figure 4: Convolutional Neural Network(CNN)

guishing more prominent segments of the picture. A feature detector, also known as a kernel or filter, is also included in our system; it scans the fields that receive an image to assess if the feature is present. This is the convolution procedure. A part of the image is first subjected to the filter and the computed dot product is then delivered to an output array. The filter continues to move forward one step at a time till the kernel has completely covered the image.A convolved feature or activation map is the collection of dot products produced by the filter and the input. After each convolutional step in a CNN, a Rectified Linear Unit (ReLU), which raises the nonlinearity[16] of the model, is applied.



Pixel Values

1	0	4	2	125	67
8	2	5	4	34	12
20	13	25	15	240	2
76	8	6	6	100	76
34	66	134	223	201	3
255	123	89	55	32	2

Kernel 3 x 3 Pixels

1	2	1
2	4	2
1	2	1

Convoluted Image

		198			

2 x 1 = 2
2 x 5 = 10
1 x 4 = 4
2 x 13 = 26
25 x 4 = 100
15 x 2 = 30
8 x 1 = 8
6 x 2 = 12
6 x 1 = 6

2 + 10 + 4 + 26 + 100 + 30 + 8 + 12 + 6 = 198

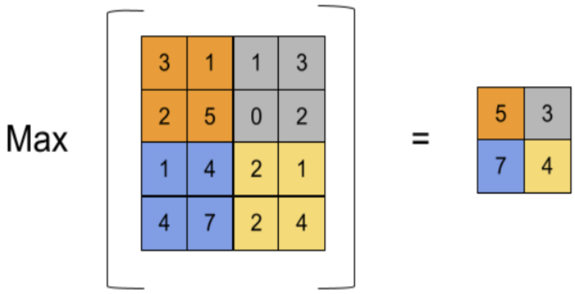
Figure 5: Convolution Layer

2.4.2 Pooling Layer

Downsampling, also known as pooling layers, reduces the number of parameters in the input and does dimensionality reduction. Although the pooling layer causes CNN to lose a lot of information, it also has significant benefits. They reduce the risk of overfitting, boost effectiveness, and reduce complexity. There are two different kinds of pooling

1. Max Pooling
2. Average Pooling

Max Pooling: As the name implies, max pooling retains the most noticeable elements in the feature map. This technique is useful for extracting the prominent or very significant aspects from an image.

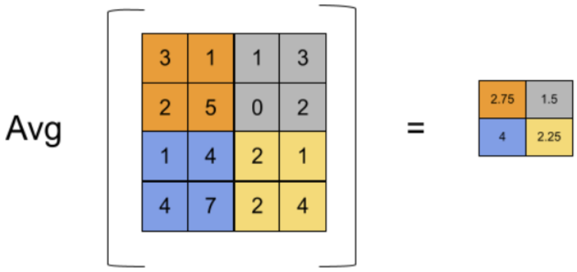


3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

5	3
7	4

Figure 6: Max Pooling

Average Pooling: As the name implies, the average pooling retains the average values of the pixel.



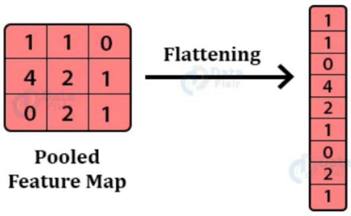
3	1	1	3
2	5	0	2
1	4	2	1
4	7	2	4

2.75	1.5
4	2.25

Figure 7: Average Pooling

2.4.3 Filter

The flattening stage, converts the pooled feature map produced during the pooling process into a one-dimensional vector.



1	1	0
4	2	1
0	2	1

1
1
0
4
2
1
0
2
1

Figure 8: Flattening Process

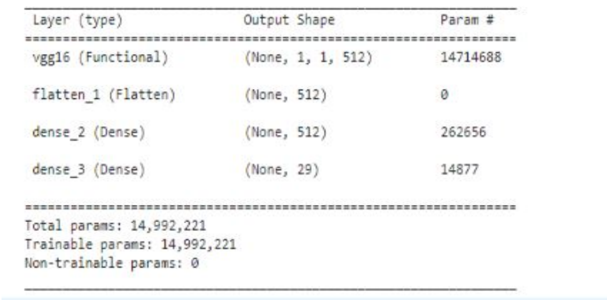
2.4.4 Fully Connected Layer

This layer performs the classification operation using the features recovered from the preceding layers and their associated filters. Convolutional and pooling layers typically use ReLu[17] functions, whereas Fully Connected layers frequently use soft-max activation functions to classify inputs appropriately.

2.5 Architectures

2.5.1 VGG16 (Visual Geometry Group)

One of the most liked and well-known algorithms for the categorization of object detection is VGG16. This architecture produced an accuracy of 92.7% while categorizing 1,000 photos into 1000 different groups. Transfer learning, a well-liked technique for categorizing photos, making it simple to apply. Utilizing VGG16, as demonstrated in Figure 9.



```
Layer (type)                Output Shape              Param #
=================================================================
vgg16 (Functional)          (None, 1, 1, 512)         14714688

flatten_1 (Flatten)         (None, 512)               0

dense_2 (Dense)             (None, 512)               262656

dense_3 (Dense)             (None, 29)                14877

=================================================================
Total params: 14,992,221
Trainable params: 14,992,221
Non-trainable params: 0
_________________________________________________________________
```

Figure 9: Implementation of VGG16

Convolution filters of varying sizes make up a VGG network. There are 13 convolutional layers in total with three completely connected layers in VGG16. An overview of the VGG architecture is provided below:
1. An image of size 224x224 is fed into VGGNet.
2. The smallest 33 receptive field is used by the convolutional filters of the VGG algorithm. A 1 x 1 convolution filter is also used by VGG to linearly transform the input.
3. Given the quick increase in the number of possible filters from 64 to 128, 256, and finally 512 in the last layers, pooling is essential.
4. Three completely interconnected layers make up VGGNet. Each of the first two layers contains 4096 channels, while the third layer contains 1000 channels—one for each class.

2.5.2 AlexNet

CNN's AlexNet was the first to deploy GPU technology to improve performance.

AlexNet Architecture:
Five convolutional layers, three max-pooling layers, two normalization layers, two fully connected layers, and one SoftMax layer make up the AlexNet architecture.

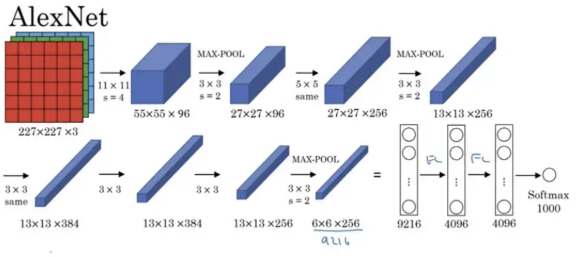


Figure 10: AlexNet

2.5.3 DenseNet

By altering the typical CNN architecture and streamlining the connectivity structure across layers, DenseNets[18] alleviate the "Vanishing Gradient Problem". A densely Connected Convolutional Network is also known as DenseNet architecture because of the direct connection between each layer and the previous layer.

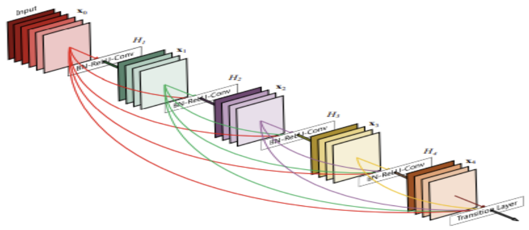


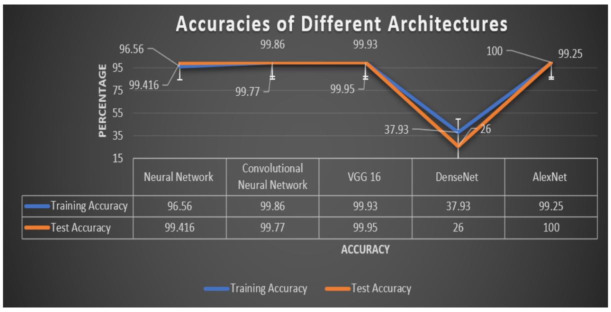
Figure 1: A 5-layer dense block with a growth rate of $k = 4$. Each layer takes all preceding feature-maps as input.

Figure 11: Block Diagram of DenseNet

DenseNet Architecture:
The four primary components of the DenseNet architecture are:
1. Connectivity
2. DenseBlocks
3. Growth Rate
4. Bottleneck layers

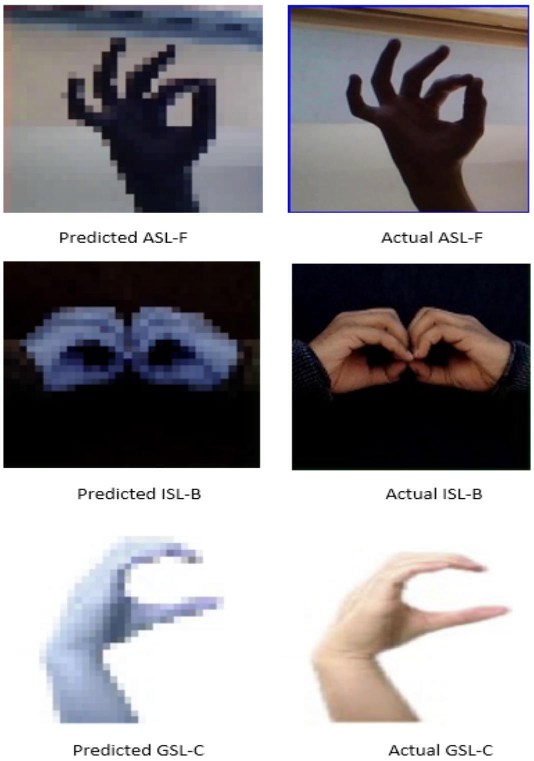
3 Experimental Results

Based on our comparative study, below is the graph that depicts the results of various models that we have trained on.
The best performing model from the above study is Convolutional Neural Network. For predictions,



	Neural Network	Convolutional Neural Network	VGG 16	DenseNet	AlexNet
Training Accuracy	96.56	99.86	99.93	37.93	99.25
Test Accuracy	99.416	99.77	99.95	26	100

Figure 12: Table and Graph Representation of Obtained Results


Predicted ASL-F

Actual ASL-F

Predicted ISL-B

Actual ISL-B

Predicted GSL-C

Actual GSL-C

Figure 13: Actual and Predicted Images

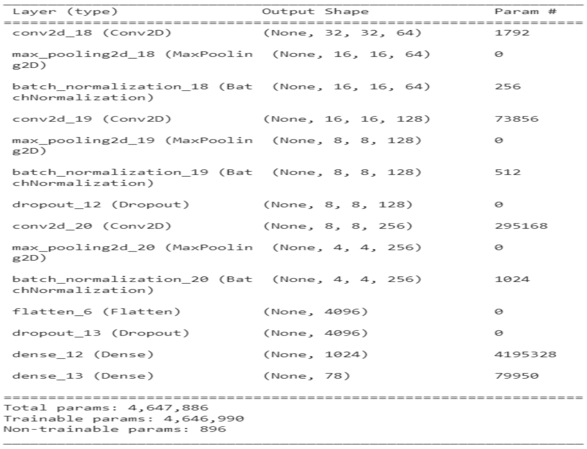
we did some minor modifications in the architecture as shown in the Figure 14.

Also, as the application part, we have created a Multilingual Sign Language translator, Figure 13 shows some of the examples of the predicted images. Given an image of one sign language gesture from our dataset, we can also convert the anticipated image to other sign language gestures.

4 Conclusion

In conclusion, we were successful in creating a system that can comprehend sign language and convert it to matching text. There are still a lot of gaps in our system, such as the fact that it can only recognize hand motions for the letters A to Z, leaving out body gestures and other dynamic gestures.

It can be further enhanced and optimized in the future.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 32, 32, 64)        1792

 max_pooling2d_18 (MaxPoolin  (None, 16, 16, 64)       0
 g2D)

 batch_normalization_18 (Bat  (None, 16, 16, 64)       256
 chNormalization)

 conv2d_19 (Conv2D)          (None, 16, 16, 128)       73856

 max_pooling2d_19 (MaxPoolin  (None, 8, 8, 128)        0
 g2D)

 batch_normalization_19 (Bat  (None, 8, 8, 128)        512
 chNormalization)

 dropout_12 (Dropout)        (None, 8, 8, 128)         0

 conv2d_20 (Conv2D)          (None, 8, 8, 256)         295168

 max_pooling2d_20 (MaxPoolin  (None, 4, 4, 256)        0
 g2D)

 batch_normalization_20 (Bat  (None, 4, 4, 256)        1024
 chNormalization)

 flatten_6 (Flatten)         (None, 4096)              0

 dropout_13 (Dropout)        (None, 4096)              0

 dense_12 (Dense)            (None, 1024)              4195328

 dense_13 (Dense)            (None, 78)                79950

=================================================================
Total params: 4,647,886
Trainable params: 4,646,990
Non-trainable params: 896
_________________________________________________________________
```

Figure 14: Modified CNN Architecture for Multi-Lingual Translator

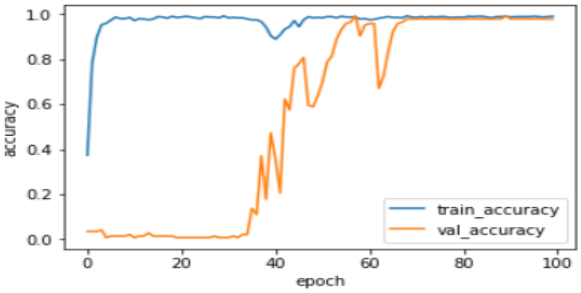


Figure 15: Epoch Visualization

References

[1] Lean Karlo S Tolentino, RO Serfa Juan, August C Thio-ac, Maria Abigail B Pamahoy, Joni Rose R Forteza, and Xavier Jet O Garcia. Static sign language recognition using deep learning. *Int. J. Mach. Learn. Comput*, 9(6):821–827, 2019.

[2] Muhammad AL-Qurishi, Thariq Khalid, and Riad Souissi. Deep learning for sign language recognition: Current techniques, benchmarks, and open issues. *IEEE Access*, PP:1–1, 09 2021.

[3] Aeshita Mathur, Deepanshu Singh, and Rita Chhikara. Recognition of american sign language using deep learning. In *2021 International Conference on Industrial Electronics Research and Applications (ICIERA)*, pages 1–5, 2021.

[4] Kshitij Bantupalli and Ying Xie. American sign language recognition using deep learning and computer vision. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 4896–4899, 2018.

[5] Shruti Mohanty, Supriya Prasad, Tanvi Sinha, and B. Niranjana Krupa. German sign language translation using 3d hand pose estimation and deep learning. In *2020 IEEE REGION 10 CONFERENCE (TENCON)*, pages 773–778, 2020.

[6] Pratik Likhar, Neel Kamal Bhagat, and Rathna G N. Deep learning methods for indian sign language recognition. In *2020 IEEE 10th International Conference on Consumer Electronics (ICCE-Berlin)*, pages 1–6, 2020.

[7] Mohammad Rostami, Bahman Bahmani, Vahid Bakhtyari, and Guita Movallali. Depression and deaf adolescents: a review. *Iranian Rehabilitation Journal*, 12(1):43–53, 2014.

[8] Necati Cihan Camgoz, Simon Hadfield, Oscar Koller, Hermann Ney, and Richard Bowden. Neural sign language translation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[9] Ankita Wadhawan and Parteek Kumar. Deep learning-based sign language recognition system for static signs. *Neural computing and applications*, 32(12):7957–7968, 2020.

[10] Ilias Papastratis, Christos Chatzikonstantinou, Dimitrios Konstantinidis, Kosmas Dimitropoulos, and Petros Daras. Artificial intelligence technologies for sign language. *Sensors*, 21(17):5843, 2021.

[11] R Martin McGuire, Jose Hernandez-Rebollar, Thad Starner, Valerie Henderson, Helene Brashear, and Danielle S Ross. Towards a one-way american sign language translator. In *Sixth IEEE International Conference on Automatic Face and Gesture Recognition, 2004. Proceedings.*, pages 620–625. IEEE, 2004.

[12] Salvador García, Sergio Ramírez-Gallego, Julián Luengo, José Manuel Benítez, and Francisco Herrera. Big data preprocessing: methods and prospects. *Big Data Analytics*, 1(1):1–22, 2016.

[13] G. Anantha Rao, K. Syamala, P. V. V. Kishore, and A. S. C. S. Sastry. Deep convolutional neural networks for sign language recognition. In *2018 Conference on Signal Processing And Communication Engineering Systems (SPACES)*, pages 194–197, 2018.

[14] Andrej Krenker, Janez Bešter, and Andrej Kos. Introduction to the artificial neural networks. *Artificial Neural Networks: Methodological Advances and Biomedical Applications. InTech*, pages 1–18, 2011.

[15] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In Lourdes Agapito, Michael M. Bronstein, and Carsten Rother, editors, *Computer Vision - ECCV 2014 Workshops*, pages 572–578, Cham, 2015. Springer International Publishing.

[16] Ruey S Tsay. Nonlinearity tests for time series. *Biometrika*, 73(2):461–466, 1986.

[17] Yu-Dong Zhang, Chichun Pan, Junding Sun, and Chaosheng Tang. Multiple sclerosis identification by convolutional neural network with dropout and parametric relu. *Journal of computational science*, 28:1–10, 2018.

[18] Rangel Daroya, Daryl Peralta, and Prospero Naval. Alphabet sign language image classification using deep learning. In *TENCON 2018 - 2018 IEEE Region 10 Conference*, pages 0646–0650, 2018.