A Deep learning approach to classifying American Sign Language

Rafi Khandoker, BRACU21, BUET22

***Abstract*—As we progress in science, we come to find the daunting challenge of the increasing number of people with hearing issues. Sign language is one of many ways to communicate with them. Hence, we present a deep learning model that utilizes convolutional neural networks to classify different American Sign Language (ASL) characters from images. Convolutional Neural Network (CNN) was used to train the dataset with three thousand images per sign. Results showed that the suggested model quite accurately around 99% distinguishes between the characters. With the help of our model, we can better interact with them.**

Keywords—Convolutional Neural Networks, American Sign Language

I. Introduction (*Heading 1*)

Over 5% of the world's population, or 430 million people, require treatment for "disabling" hearing loss (432 million adults and 34 million children). By 2050, around 700 million people, or one in ten, will have significant hearing loss.[1]

Hearing loss in the better-hearing ear above 35 decibels (dB) is deemed debilitating. Nearly eighty percent of individuals with hearing impairment reside in low- and middle-income nations. As people age, hearing loss becomes more widespread; more than a quarter of persons over the age of 60 have significant hearing loss.[1]

Up to ninety percent of deaf children are born to hearing parents, making sign language acquisition a family concern. Parents who study American Sign Language with their deaf children can often communicate with their children on a deeper level. When deaf or hard-of-hearing youngsters learn American Sign Language, their ability to learn their native language increases, according to research. Similarly, lipreading takes repeated practice.

Our main objective of this paper is to help the children and the adults to learn and teach ASL better and easier. To achieve this goal, we propose a system consisting of a CNN model.

Images are the ideal form of machine learning data: simple for people to comprehend yet challenging for computers to process. Image analysis has aided in the creation of deep neural networks for applications such as identifying vehicles, humans or animals, lanes, pedestrians, and so on. A new approach for the technology is to predict sign languages for broader communication between people.

Recently, significant progress has been made in developing computer vision systems that transform sign language into spoken English. This method usually employs powerful neural network architectures that are capable of identifying minute patterns in streaming video. However, as a first step in learning how to design a translation system, we may reduce the scale of the issue by translating individual letters rather than words.

For training, we chose a dataset [2] that had around 3000 images per category we wanted to classify. We introduced all the 26 alphabets of English language as well as 2 additional characters of the American Sign Language (ASL) and one empty background class for better identification.

We chose a specific category like nothing with only the background image to keep overfitting in check.

II. Literature Review

In this paper a systematic review of around 300 published papers on sign language recognition, with over 400 experimental results. It includes the vast majority of papers published between the field's beginnings in 1983 and the year 2020. It also offers a fine-grained comparison of over 25 research projects' identification algorithms on RWTH-PHOENIX-Weather 2014, the industry's standard benchmark task. Sign language recognition research has evolved significantly in the last decade, receiving more attention than ever. This study outlines the state of the art to aid in the development of the field and to raise unsolved problems.[3]

This paper proposes a Leap Motion Controller-based system for recognizing the American Sign

Language alphabet (LMC). 2012 marked the introduction of the LMC hand skeleton data provider, which has been in use since since. The proposed model is evaluated through experimentation using a dataset created for this purpose. Because the hand region is illuminated by the IR LEDs, the undistorted images are processed with a brightness threshold and cropped to eliminate the backdrop.

The recovered features are then fed to a neural network classifier to get an accurate forecast. By combining a 16-dimensional vector and a 144-dimensional vector produced by the skeleton module, a 128-dimensional image is produced. When 24 neuronal layers and SoftMax methods are applied to each image, the primary output and SM output are the results. The leave-one-and-a-half-half test results for each person vary. The subject with the highest prediction accuracy is the fourth one, with a score of 91.2%, which is 23% higher than the lowest score from the third and fourth subjects.

Incorrect classifications begin with "C-E," "K-P," "R-U," "S-N," "T-S," and "X-D." The 'N-S' combination is the most puzzling of all possible pairings (380 instances of 'N' were mislabeled as 'S'). [4] From this study we learned that we must be aware of letter misdetections and thus we tried to incorporate a confusion matrix to better distinguish the issue.

This research provides a model for recognizing and categorizing hand movements based on a convolutional neural network. The dataset includes 26 different hand movements corresponding to the 26 letters of the English alphabet. The Hand Gesture Recognition dataset, which can be obtained from the Kaggle website, was utilized in this work. The collection has 27,455 photographs showing different hand gestures. CNN-based deep learning is used to automatically extract and learn motion detection information. This analysis compares and analyzes four modern studies. The proposed approach ensures a 99 percent test accuracy. [5]

Bhavana et al. (2021) sought to construct a model that can comprehend human speech and gestures. To increase communication between hearing people and people with cognitive impairments, they developed a model that could translate sign language and another model that could detect simple motions. Several techniques, such as KNN, Logistic Regression, Nave Bayes Classification, Support Vector Machines (SVM), and Convolutional Neural Networks, can be used to perform the problem (CNN). CNN was chosen over the alternatives because it produces more accurate results. Using custom-built Python software, they employ the CNN method to train the model. [6]

The authors of this paper are affiliated with the Computer Science and Engineering Departments at both the University of Aizu and the University of Asia Pacific in Dhaka, Bangladesh. Human-computer interaction refers to interactions between humans and machines (HCI). Using HCI, individuals can interact with computers in novel ways. New technologies, such as Direct Neural Interfaces (DNI), are implemented in HCI.

Digital neural interfaces (DNI) enable machines to detect brain activity in the absence of a person. Human hand gestures are utilized in several industries, including computer gaming, virtual reality, and sign language detection. Many academics are striving to increase the efficiency, usefulness, and sturdiness of human touch interfaces. The paper was presented at the 4th International Conference on Frontiers of Signal Processing. As Modanwal and his colleagues discovered, it is possible to overcome the gap between humans and computers using gesture recognition. Liang employed a Hidden Markov Model (HMM) for sign language recognition. This is the operation of the proposed model, as depicted in Figure 1. Figures 2 and 3 depict the preprocessing of gestural images to determine the region of interest. Figure 4 displays the ASL hand motions of three different people using 26 letter signs in three distinct ways.

In this paper the authors provided a system for non-contact ASL hand gesture identification. Using an HCI, the SLR of American Sign Language (ASL) for those with disabilities can be generated with an impressive 94.57% accuracy. [7]

Imagine your eyes gently gliding across a printed page, like a flashlight illuminating the upper left corner of the image. This is the most effective approach to comprehend convolutional layers. By tracing the kernel (a sliding window) through the image, convolutional layers perform element-wise matrix multiplication between the values of the kernel and the values of the pixels.

Currently, human-robot interaction relies mainly on hand gesture recognition based on vision (HRI). Due to this non-contact technology, human-robot interactions are more natural and enjoyable than ever before. In order to recognize ASL movements, we construct a two-stream convolutional neural network architecture (2S-CNN) that combines data from various channels (RGB and depth). We should begin by eliminating noise and background motion from the hand gesture data. In the second step of identifying hand motions, convolutional neural networks are used to extract RGB and depth information from two streams. A fusion layer is then constructed to merge the recognition results of the two streams. This technique employs multimodal data to enhance the accuracy with which ASL hand motions are recognized. [8]

The American Sign Language (ASL) alphabet can now be recognized using a revolutionary method developed by Microsoft's Kinect. The approach can

identify 24 ASL alphabet signs with an accuracy rate of above 90% when compared to older benchmarks. Utilizing publicly available data from Surrey University, the results have been independently confirmed. The Joint Localization Algorithm restricts joint placements based on the hierarchy of the hand.

A kinematic constraining technique is used to overcome the problem of localizing the hand's position. The Kinect sensor used 72,000 images to record the depth of a hand. The accuracy of the dataset with 10 million pixels was 88.96% while using distance adaptive scheme (DAS), as opposed to 81.34% when using other methods. The system had an average accuracy of 92% on a dataset of 24 static letter signs. The technique can also be applied to other hand-based applications, like remote industrial robot control or remote communication with medical assistants from a hospital ward. [9]

Hand signals and signs are used extensively by those who are deaf or have difficulties speaking. Human-computer interaction (HCI) technologies allow us to communicate with computers using hand gestures. In mainstream sign languages, different decimal places are represented by hand movements. Recent breakthroughs in Deep Convolutional Neural Networks have made it possible to recognize and classify hand gestures with greater precision (DCNN). In this study, a layer-wise optimum neural network architecture is provided by using batch normalization to speed up training convergence and dropout to reduce data overfitting. When batches are normalized to a mean zero and a variance one, the model's gradient flow improves and converges faster. Overfitting is reduced by making neurons more regular through dropout. A numerical hand motion data set was created to verify the assertions based on American Sign Language. [10]

The authors employed the segmentation of depth images to extract the gesture region from the image. Training Convolutional Neural Networks (CNNs) and support vector machines helps to extract features and recognize motions (SVMs). The gesture region can be segmented via one of three techniques: segmentation based on 2D forms, skin color, or depth information. [11]

Computer vision has difficulty recognizing ASL alphabets because of signal complexity, high interclass similarity and intraclass broad diversity, and frequent occlusions. As we have seen from previous discussions many characters and hand gestures from a different angle in the picture does make it difficult for CNN to classify correctly. In this study the authors combined CNN with Multiview augmentation and inference fusion to differentiate ASL alphabets from depth images received from Microsoft Kinect. When they employ their technique, their augment to their original data with new perspectives, which improves training and minimizes the chance of overfitting. The inference phase of their system analyzes data from several perspectives to solve challenging scenarios caused by orientational fluctuations and partial occlusions. Their method exceeds the competition on two publicly available benchmark datasets. [12]

III. PROPOSED MODEL

The working flow of the model is explained sequentially in the following parts.

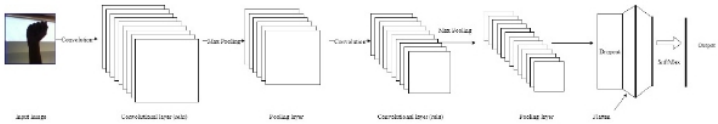

Fig 1. Workflow diagram

The dataset [2] used had 87000 images and all were divided among 29 (26 alphabets of the English language and 2 asl characters and one nothing class) classes equally. Each layer of the CNN model has distinct activation functions, such as ReLu and Sigmoid, to verify the picture as output in the relevant layer. The number of layers used are explained below.

A. Dataset Collection

The dataset used in this research has 87000 images of sign languages containing 29 different classes(all 26 alphabets, delete, space and nothing). All 29 classes had 3000 images to be trained on.

Fig 2. All 29 characters

We split the dataset using the train-test split where we trained upon 90% of the images and tested on 10% of the images.

```
Total number of symbols:  29
Number of training images:  78300
Number of testing images:  8700
Number of evaluation images:  870
```

Fig 3. Dataset summary

B. *Data Processing*

Because the data is scattered across several directories, it must be retrieved and saved in arrays. Since I will execute this technique twice, once on the training data and again on the assessment data, it makes sense to construct a function to facilitate the procedure. This function adds each image to an array after scaling it to the appropriate dimensions which in our case keeping the original image ratio and converting into 64 x 64 pixels. In addition, it adds the relevant label code to a collection of labels. A label code is only a number that is assigned to each label. For instance, A equals 0, B equals 1, C equals 2, etc. This method is known as label encoding.

The image labels were encoded with one-hot encoding, described below, for quicker and more accurate predictions.

For our dataset label encoding, A is encoded as 0, B as 1, C as 2, D as 3, etc., whereas nothing is encoded as 28.

Therefore, our letter labels are now encoded as categorical integers, with 'A,' 'B,' and 'C' respectively encoded as 0, 1, and 2. The labels must be one-hot encoded before being presented to a Keras model.

This converter will increase the dimension of the array of labels from one to two. Each row of the array of one-hot encoded labels in two dimensions corresponds to a unique label.

1) *One hot encoding:*

One-hot encoding is the process of transforming categorical data into a format that may be given to machine learning algorithms to increase prediction accuracy.

One-hot encoding is a common strategy in machine learning for dealing with categorical data. Categorical variables must be modified in the pre-processing step since machine learning models demand numeric input variables. Nominal and ordinal data can be found in categorical data.

For each unique value, this approach adds a new column to the current category column. These fictitious variables are subsequently filled with 0s and 1s (1 meaning TRUE, 0 meaning FALSE). [13]

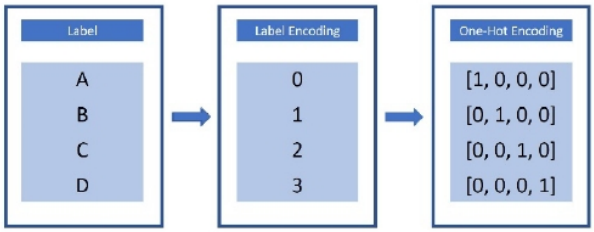

Fig 4. Example of One Hot Encoding

2) *Normalizing RGB values*

Each image is made up of three parts: one for each of the Red, Green, and Blue (RGB) channels. The component values are stored as integer numbers ranging from 0 to 255, the range provided by a single 8-bit byte.

If we divide the range by 255, we get 0.0-1.0, where 0.0 represents 0 (0x00 in hex) and 1.0 represents 255. (0xFF in hex). Normalization will help us remove image distortions caused by lighting and shadows. Because our images have a diversity of light and shadow areas, this is an ideal technique for our dataset. Following the initial scaling of the picture, the next section comprises the (Blue, Green and Red) BGR to RGB image color space transition. Because of OpenCV's image importing feature, this method is used in this model. Because of its unique importing capabilities, OpenCV usually imports photos in BGR format. As the conversion method merely transfers values from one array to another, it was verified for each picture to see whether the conversion was successful.

IV. Model Description

Convolutional Neural Network

A convolutional neural network (CNN or ConvNet) is a form of artificial neural network (ANN) that is frequently used in deep learning to evaluate visual data. Due to their shared-weight construction of convolution kernels or filters that slide along input features and generate translation-equivariant feature maps as outputs, CNNs are often referred to as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN).

These are not, however, the only layers necessary to achieve our task. The whole neural network architecture will consist of several layers, number of layers and each layer functions are described below, each of which will play a specific function in the network's overall operation.

For the proposed CNN model, we used 5 conv2d layers, and 2 maxpooling2d layers, to avoid overfitting

including 3 dropout layers that dropped around 50% and used a flatten layer right before the final dense layer where we classified 29 (26 alphabets of the English language and 2 asl characters and one nothing class) characters. For our activation function, we used SoftMax activation function described below as well. For the pooling layers we used 4, 4 strides and for conv2d layers, we used Relu (Rectified linear unit) described below.

A. Conv2d:

This layer produces a convolution kernel which is convolved with its input to yield a tensor of outputs. If the employ bias parameter is set to True, a bias vector is appended to the outputs. [14]

B. Maxpooling2d:

The input is down-sampled along its spatial dimensions (height and width) by calculating the maximum value of each input channel over an input window (of a size determined by pool size). Using strides, each window dimension is adjusted. [15]

In our model, this layer slides a sliding window (4, 4) across the image and downscales it by displaying the highest value within the window.

C. Dropout:

Randomly shifting a percentage of the input units to 0 during training prevents the model from overfitting.

D. Flatten:

This turns the output of the convolutional component of the CNN into a one-dimensional feature vector that may be passed to the subsequent fully connected layer.

E. Softmax activation function:

SoftMax returns the related probability for each class. Here is the equation for the SoftMax activation function.

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{1}$$

Here, Z indicates the values of the neurons in the output layer, exp(zi) represents a standard exponential function for the input vector and exp(zj) represents a standard exponential function for the output vector. These values are subsequently divided by the sum of exponential values to normalize and convert them into probabilities. Assume that the values of classes Z21, Z22, and Z23 are 2.33, -1.46, and 0.56, respectively, for multiclass classification problems such as this one. These values are the result of applying the SoftMax activation function to each of these neurons.

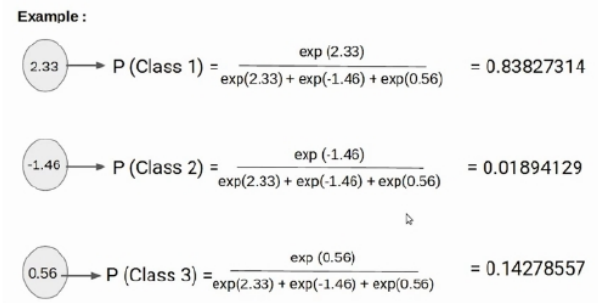

Fig 5. SoftMax function example

Note that we are assuming that there is a total of 3 classes and therefore the sum of all 3 possibilities would be equal to 1. So, if the likelihood of any of these classes is changed, the probability value of the first class would likewise change. [16]

So, for our multi-class classifier, the SoftMax functions equation would be

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2}$$

Here, σ represents the SoftMax function, z represents the input vector, e^{zi} represents a standard exponential function for the input vector, K represents several classes in the multi-class classifier, and e^{zj} represents a standard exponential function for the output vector.

F. Dense:

In a completely linked layer, every input is connected to every output.

To check how everything balances up, we printed the model summary. Our model has a massive 1,660,253 parameters. These are the various weights that the model learned during training and uses to generate predictions on a new image.

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 (MaxPooling2	(None, 4, 4, 128)	0
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

Fig 6. Model summary

V. RESULTS

After training the model on the dataset we reached an astounding accuracy of around 99% on our test images.

A confusion matrix is a table that displays how well a classification model (or "classifier") performs on a set of test data with true values that are known. The rows of the matrix represent examples from the actual class, while each column represents an occurrence from the forecasting class (or vice versa). The name refers to how simple it is to determine whether the system is combining two distinct kinds of data (i.e., commonly mislabeling one as another).

Here is a confusion matrix where we plot true labels vs predicted labels on our training images.

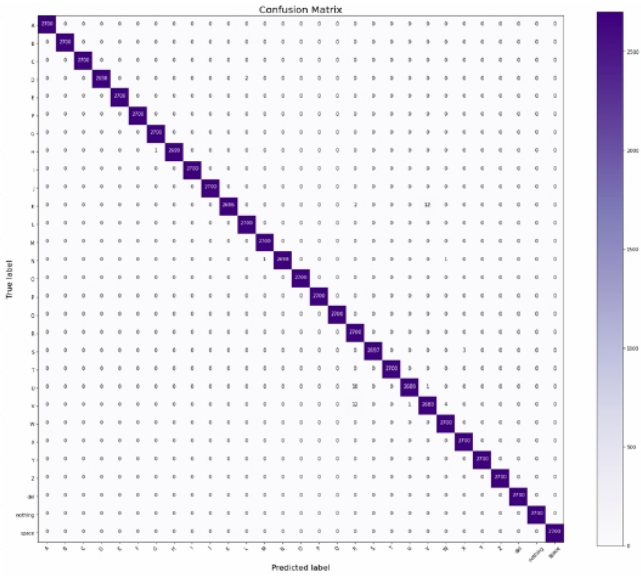

Fig 7. Confusion matrix of training data (True label vs Predicted label)

Here we can see how our model performed for each class in the training session. We can see which characters were missed how many times and which character was falsely labeled.

Below is another confusion matrix but here we plot which showed the true labels vs the predicted labels on our test images.

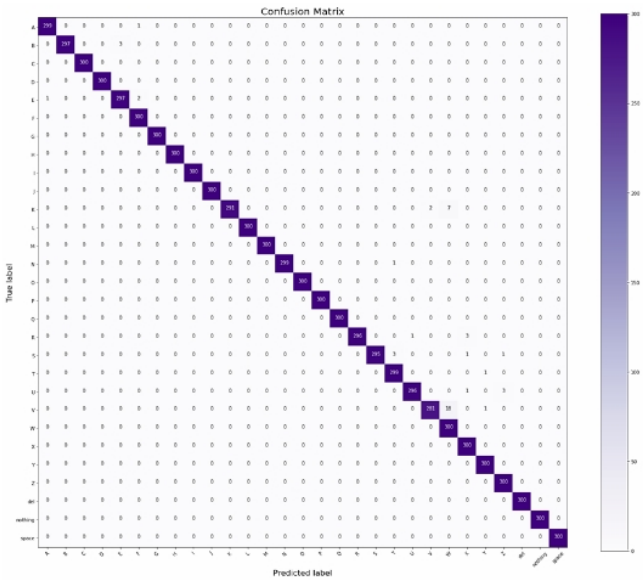

Fig 8. Confusion matrix of testing data (True label vs Predicted label)

Here we can see how our model performed for each class in the test images. We can see which characters were missed how many times and which character was falsely labeled.

VI. FUTURE WORK

Although we achieved high accuracy on our model there were still a few improvements that could be done for the future work

1. We could implement the same model on a continuous image dataset
2. We could implement the model in a server to give predictions in real time

ACKNOWLEDGMENT

First and foremost, we would want to thank Allah for allowing us to complete this job without any significant problems. Next, we would want to thank our parents, who have been our biggest supporters in reaching our current achievement. In addition, we would like to express our gratitude to our outstanding Supervisor, who aided us by providing continuous feedback and suggestions for enhancing our work and accomplishing our ultimate target.

REFERENCES

[1] Deafness and hearing loss. (2022). Retrieved 17 August 2022, from https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss

[2] ASL Alphabet. (2022). Retrieved 19 July 2022, from https://www.kaggle.com/datasets/grassknoted/asl-alphabet

[3] Koller, O. (2022). Quantitative Survey of the State of the Art in Sign Language Recognition. Retrieved 31 July 2022, from https://arxiv.org/abs/2008.09918v2

[4] *Par.Nsf.Gov*, 2022, https://par.nsf.gov/servlets/purl/10083240.

[5] Goswami, T., & Javaji, S. (2020). CNN Model for American Sign Language Recognition. *Lecture Notes In Electrical Engineering*, 55-61. doi: 10.1007/978-981-15-7961-5_6.

[6] D. Bhavana, K. Kishore Kumar, Medasani Bipin Chandra, P.V. Sai Krishna Bhargav, D. Joy Sanjana, G. Mohan Gopi. Hand Sign Recognition using CNN. *International Journal of Performance Analysis in Sport*, 2021, 17(3): 314-321 doi:10.23940/ijpe.21.03.p7.314321.

[7] Ma, Miao et al. "A Recognition Method Of Hand Gesture With CNN-SVM Model". *Bio-Inspired Computing – Theories And Applications*, 2016, pp. 399-404. *Springer Singapore*, https://doi.org/10.1007/978-981-10-3611-8_33. Accessed 1 Aug 2022.

[8] Gao, Q., Ogenyi, U., Liu, J., Ju, Z., & Liu, H. (2019). A Two-Stream CNN Framework for American Sign Language Recognition Based on Multimodal Data Fusion. *Advances In Intelligent Systems And Computing*, 107-118. doi: 10.1007/978-3-030-29933-0_9

[9] Cao Dong, Ming C. Leu and Zhaozheng Yin. American Sign Language Alphabet Recognition Using Microsoft Kinect. In IEEE International Conference on Computer Vision and Pattern Recognition Workshops, 2015.

[10] A. K. Tushar, A. Ashiquzzaman and M. R. Islam, "Faster convergence and reduction of overfitting in numerical hand sign recognition using DCNN," *2017 IEEE Region 10 Humanitarian Technology Conference (R10-HTC)*, 2017, pp. 638-641, doi: 10.1109/R10-HTC.2017.8289040.

[11] Tao, Wenjin et al. "American Sign Language Alphabet Recognition Using Convolutional Neural Networks With Multiview Augmentation And Inference Fusion". *Engineering Applications Of Artificial Intelligence*, vol 76, 2018, pp. 202-213. *Elsevier BV*, https://doi.org/10.1016/j.engappai.2018.09.006. Accessed 1 Aug 2022.

[12] M. R. Islam, U. K. Mitu, R. A. Bhuiyan and J. Shin, "Hand Gesture Feature Extraction Using Deep Convolutional Neural Network for Recognizing American Sign Language," *2018 4th International Conference on Frontiers of Signal Processing (ICFSP)*, 2018, pp. 115-119, doi: 10.1109/ICFSP.2018.8552044.

[13] *What is One-hot Encoding - Deepchecks. (2022). Retrieved 31 July 2022, from https://deepchecks.com/glossary/one-hot-encoding/#:~:text=One%2Dhot%20encoding%20in%20machine,categorical%20data%20in%20machine%20learning.*

[14] *Team, K. (2022). Keras documentation: Conv2D layer. Retrieved 31 July 2022, from https://keras.io/api/layers/convolution_layers/convolution2d/#:~:text=Conv2D%20class&text=2D%20convolution%20layer%20(e.g.%20spatial,and%20added%20to%20the%20outputs.*

[15] *Team, K. (2022). Keras documentation: MaxPooling2D layer. Retrieved 31 July 2022, from https://keras.io/api/layers/pooling_layers/max_pooling2d/#:~:text=MaxPooling2D%20class&text=Max%20pooling%20operation%20for%202D,by%20strides%20along%20each%20dimension.*

[16] *Softmax | What is Softmax Activation Function | Introduction to Softmax. (2022). Retrieved 31 July 2022, from https://www.analyticsvidhya.com/blog/2021/04/introduction-to-softmax-for-neural-network/*