Development of a Sign Language Recognition System Using Machine Learning

Hope Orovwode
Electrical and Information Engineering
Covenant University
Ota, Ogun State, Nigeria
hope.orovwode@covenantuniversity.edu.ng

Ibukun Deborah Oduntan
Electrical and Information Engineering
Covenant University
Ota, Ogun State, Nigeria
oduntan.Ibukunoluwa@stu.cu.edu.ng

John Abubakar
Electrical and Information Engineering
Covenant University
Ota, Ogun State, Nigeria
john.abubakar@covenantuniversity.edu.ng

2023 International Conference on Artificial Intelligence, Big Data, Computing and Data Communication Systems (icABCD) | 979-8-3503-1480-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICABCD59051.2023.10220456

***Abstract*—Deafness and voice impairment have been persistent disabilities throughout history, hindering individuals from engaging in verbal communication and leading to their isolation from the predominantly vocally communicating society. Sign language has emerged as the primary mode of communication for people with these disabilities. However, it presents a language barrier as it is not commonly understood by those who can hear. To address this issue, various methods for recognizing sign language have been proposed. This paperaims to develop a machine learning-based system that can recognize sign language in real-time. The paper involved the acquisition of a dataset consisting of 44,654 images representing the static American Sign Language (ASL) alphabet signs. The HandDetector module was utilized to detect and capture images of the signer's hand forming each sign through a PC webcam. The dataset was split into three sets: training data (20,772 cases), validation data (8,903 cases), and test data (14,979 cases). Image pre-processing techniques were implemented on the images and a convolutional neural network (CNN) model was trained and compiled. The CNN utilized in the paper comprised of three convolutional layers and a SoftMax output layer and it was compiled using the Adam optimizer and categorical cross-entropy loss function. The performance of the system was evaluated using the test dataset. Notably, the system achieved remarkable accuracy rates, having a training accuracy of 99.86%, a validation accuracy of 99.94%, and a test accuracy of 94.68%. The results obtained from this study demonstrated significant advancements in sign language recognition, surpassing previous findings in the literature.**

Keywords—machine learning, sign language recognition, CNN, sign language, American Sign Language (ASL)

I. INTRODUCTION

Based on a World Health Organization (WHO) report, greater than 6% of the general population has hearing impairment while approximately 5% of the population of the world, or 430 million people (432 million adults and 34 million children), need therapy to treat their 'disabling' hearing loss [1]. Debilitating hearing loss is projected to affect about 700 million individuals by 2050, or one in ten people [2]. Treat (2016) estimates that 23.7% of Nigerians, out of a population of over 155 million people, have a hearing disability (total deafness, hearing loss, or other impairment related to hearing) [3]. Additionally, up to 84% of Nigeria's deaf population is uneducated and underdeveloped economically [4]. WHO defines someone as having loss of hearing if they are unable to hear at the level of a person with normal hearing, which is characterized by possessing hearing levels of 20 dB or higher in both ears [5]. It can range from mild to severe and can affect either or both ears [6]. The term "deaf and dumb" is frequently used to refer to those who, due to their deafness, are unable to hear what others are saying and, as a result, continue to be dumb [7]. Children who have early hearing loss due to illness or an accident quickly lose their ability to speak [8]. This is because deaf children are unable to acquire speech by imitating others [9]. Hereditary, degenerative, and accidental factors among others can lead to hearing loss [10].

Loss of hearing may impede the capacity of individuals to express themselves verbally. This restriction makes it difficult for healthy and deaf individuals to communicate with one another. Due to this, deaf and mute people often find it difficult to have meaningful conversations with people they encounter on a regular basis. The primary method of communication for persons with trouble speaking or hearing impairments is sign language [11]. For this mode of communication to work, the regular individual must be able to understand sign language. Acquiring adequately trained translators in any of the numerous signed languages often requires much time and resources and may also hinder the privacy of the hearing- impaired person [12]. For this reason, many research papers have been carried out to develop sign recognition systems for various languages. The most effective methods for understanding sign language are through vision-based and wearable sensing methods like sensory gloves [13]. To determine the hand posture for identification, the glove- based system uses mechanical or optical sensors which are connected to the user's glove and transforms finger motions into electrical signals [14][15]. However, these devices are bulky and typically have many cables connecting to a computer. This demonstrates the necessity for non-intrusive, image-based approaches for recognizing motions [16][17]. In a vision/image- based technique, joint angles, finger positions, and characteristics relating to the palms are calculated and used to accomplish recognition [18]. With this methodology, the signs must be photographed or recorded on video, then processed using image-processing software [19][20]. The three types of image-based sign language recognition systems are alphabet, individual word, and continuous sequences [21]. Since it is quicker to use words and continuous sequences than to spell out each individual word, people who have trouble hearing or speaking most often interact with others in this way. However, signers will employ finger spelling if there is no conventional sign to represent the required word [22]. They spell out the word with hand gestures that represent the letters of the alphabet. In this instance, a static posture individually produces each letter. In certain sign languages, a

979-8-3503-1480-9/23/$31.00 ©2023 IEEE

single hand is used for finger spelling, whereas two hands are used in others [23].

There are many distinct kinds of sign language in the world including British sign language, Spanish sign language, Arabic sign language, Chinese sign language and so on [24]. The proposed sign language recognition system focuses on the American sign language and uses a vision-based approach to detect finger spelling ASL hand gestures and translate them into text. Image processing algorithms are used by the system to detect and track the movements of the hand and fingers in real-time. The system also uses machine learning algorithms to recognize the gestures and translate them into text.

In this paper, we will discuss some related works on the development of sign language recognition system. We will also explain in detail how the proposed system was developed and implemented, and the results obtained.

II. Literature Review

Numerous research studies have been conducted on sign language recognition, utilizing various sensor technologies and machine learning methods. Below is a quick rundown of some of the recent research that has been carried out on the detection and recognition of sign language.

In their study, Thakur et al. [25] proposed employing a Convolutional Neural Network (CNN) to achieve real-time detection of sign language and generate corresponding speech. The dataset used in the study mostly consisted of American Sign Language alphabet, and the preprocessed gesture dataset were trained using the CNN VGG-16 model with Python libraries and tools such as OpenCV and skimage. The system detected input and generated speech accordingly. The results depicted that the training loss and accuracy were 0.0259 and 99.65%, respectively, and 99.62% of the tests were accurate. By demonstrating the feasibility of using CNN, the study showed how real-time detection of sign language and subsequent speech generation could be achieved, potentially offering a more efficient communication method for people who are deaf or hearing-impaired. Further research can be done to broaden the application of the system to include more sign languages and gestures.

In the study conducted by Shrenika and Madhu Bala [26], the authors explored the use of a template matching algorithm for sign language recognition. The study involved recording different hand gestures using a camera and processing the images using various algorithms. The first step in the process was pre-processing the image, followed by edge detection to identify the edges of the sign. Once the sign was recognized using the template matching algorithm, the corresponding text was displayed. The system was able to successfully detect basic static hand signs, indicating that template matching can be an efficient technique for sign language recognition.

Tolentino et al. [27] conducted research on the application of deep learning techniques to recognize static sign language. The authors employed a skin-color modeling technique to isolate the pixels of the hand from the background. Afterwards, the images were subjected to classification using a CNN model, which had been trained using Keras. When adequate illumination and a consistent background were maintained, the average testing accuracy reached an impressive 93.67%. The system's accuracy in recognizing American Sign Language (ASL) letters was 90.04%, number recognition was 93.44%, and static word identification was 97.52%. The study showed that deep learning techniques can be useful for static sign language recognition and can outperform previous research in this field. The authors suggest that further research could focus on improving the system's accuracy under varying lighting conditions and complex backgrounds to make the system more robust for real-world applications.

Puri et al. [28] proposed a method for Indian Sign Language (ISL) recognition through the use of Python programming language. The program's code was written using Python, and several modules, including Numpy, Os, Tensorflow, Keras, OpenCv (Cv2), and various preprocessors were employed to train the system. To enhance accuracy, the training process utilized both a locally created library of ISL symbols like numbers 0-9, and an online database obtained from GitHub. The proposed algorithm demonstrated an accuracy range of 79 to 100%. The study demonstrated the potential of using Python-based deep learning techniques to recognize Indian Sign Language gestures accurately. The authors suggest further research in this direction could be fruitful in enhancing the system's efficiency and generalizability to various signing styles.

The study by Sharma and Kumar [29] proposed a technique called ASL-3DCNN for recognizing American Sign Language (ASL) using 3-D Convolutional Neural Networks (CNNs). To prepare the video sequences for processing, the frames were separated and then pre-processed by converting them to grayscale, filtering out noise and spots, and removing illumination variations through histogram equalization. 25 frames are then condensed and normalized before being trained on 3-D CNNs. The suggested technique outperformed current cutting-edge models in metrics including accuracy, recall, and f-measure, with a 0.19 second computation duration per frame, which makes it well suited for real-time applications.

Using machine learning techniques, Amrutha et al. [26] devised a system capable of recognizing sign language. The system's performance was assessed using a dataset comprising hand gestures representing numbers from 1 to 5. During the pre-processing phase, the background was eliminated using a threshold approach, and the outlines of the fingers were extracted using contour-based segmentation. To extract features and classify the gestures, K-NN with the convex hull method was used, and Euclidean distance was employed. The system had an accuracy rate of 65% on the test dataset, which could be improved by using a larger dataset and a different classifier.

Quin et al. [30] developed a British sign language recognition system with multi-class support vector machines (SVM) and histogram of oriented gradients (HOG) computer vision techniques. A real-world dataset of 13,066 cases was used to successfully test the system, which had an accuracy rate of approximately 99% and a 170 ms mean processing time, making it suitable for real-time visual signature.

Overall, the existing studies have made significant contributions to sign language recognition , but there is a need for further research to address the representation and recognition of a wider range of sign languages and gestures, including dynamic signing. This would ensure that the developed systems are more inclusive, applicable, and beneficial to diverse sign language communities.

III. METHODOLOGY

There are five main stages in the system's development as seen in Fig 1, which are data acquisition, image pre-processing, training of the model, model evaluation, and deployment of the model in which the model's real time prediction and results are obtained. The system's design specifications, as well as the step-by-step procedure for developing the system are explained in this chapter.

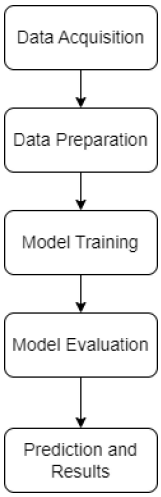


Fig. 1. Block Diagram representing the System Design

A. Data Acquisition

A collection of images depicting static ASL alphabet signs was generated to form a dataset. To acquire the image datasets, the laptop webcam was used in combination with the python OpenCV library to capture images of the signer's hand. The HandDetector module from cvzone was used to locate and crop out the signer's hand within the camera's field of view. This was done to minimize the effect of the user's background on the model prediction. Once the hand was detected, the OpenCV library was used to resize the image to a fixed size of 300 x 300 pixels. However, the dynamic signs J and Z were removed from all datasets, leaving only 24 image classes. This resulted in a total of 44654 images, which were saved in JPG format. The training dataset was divided into two separate portions - one designated for training the model, and the other reserved for validation purposes. 70% (20772 instances) was used for training while the remaining 30% (8903 instances) was used for validation. The test dataset was a separate dataset and contained 14979 instances. The distribution of the dataset is as shown in Fig 2.

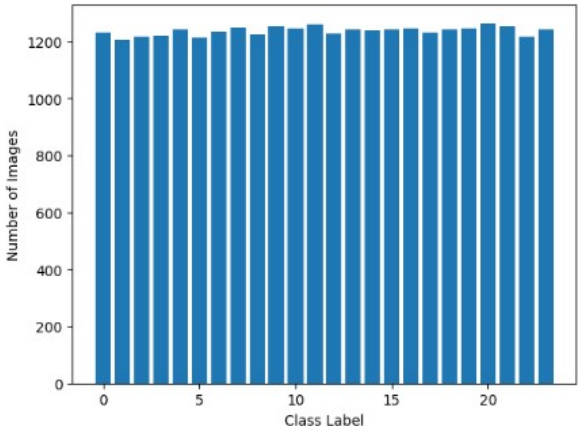


Fig. 2. Bar graph illustrating the distribution of data for each class

B. Data preparation/pre-processing

The sign images were pre-processed at this stage using image resizing and normalization techniques. In image resizing, the images were obtained in RGB format and resized to 224x224 pixels to suit the model. This was done using built-in functions from OpenCV. In the normalization step, the to_categorical function from Keras was used to convert the datasets into one-hot encoded vectors. The images were then adjusted to alter the range of pixel intensity values, yielding a mean and variance of 0 and 1, respectively. This was done by dividing the image datasets by 255. This was done to ensure that all the pixel values have a similar scale. The formula for normalization when the input data is in the 0-255 range is given in equation (1).

$$X\_norm = \left(\frac{X}{255} - \frac{X\_min}{255}\right) / \left(\frac{X\_max}{255} - \frac{X\_min}{255}\right) \quad (1)$$

C. The CNN Architecture

The model architecture was built using the Keras framework. The validation dataset was used to obtain optimal settings for the CNN hyper-parameters. The input shape of the images is specified as (224, 224, 3), indicating that the images have a resolution of 224x224 pixels and contain three color channels (Red, Green, and Blue) to represent the RGB values of each pixel. RGB images were used to provide additional color information to help the model easily distinguish between different signs and improve sign language recognition accuracy.

The model's structure comprises of three convolutional layers that each utilize Rectified Linear Unit (ReLU) activation and a max pooling layer to extract characteristics from the input images. The amount of filters in the convolutional layers progressively increases from 32 to 64 to 128. The convolutional layer outputs are then flattened and passed on to fully connected layers with ReLU activation to categorize the input images based on the extracted characteristics. The output layer has 24 neurons with Softmax activation, which generates a distribution of probabilities over the 24 classes the model is designed to classify. The model is as shown in Fig 3.

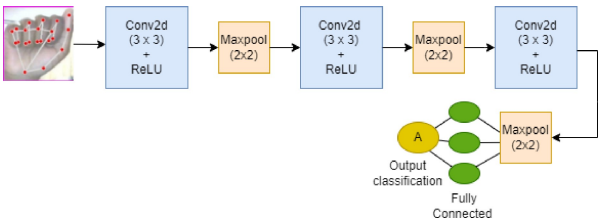


Fig. 3. CNN Model

The model was compiled using Adam optimizer once the required hyper-parameters were established with the loss function being categorical cross entropy. Updates to network weights are made using Adam. It has simple implementations, is computationally efficient, and uses less memory. This approach estimates the first and second moments of the gradient for determining the adaptive learning rate for each parameter given a set of different parameters. The layers of the model are further analyzed as thus:

1) *Input layer:* This should contain the image data.
2) *The 2D convolution layer (Conv2d):* applies a kernel that convolves with the input of the layer, resulting in a tensor of outputs.. The 3x3 kernel which makes up the

convolution layer is a convolution matrix that, when convolved with an image, performs several different tasks, including blurring, edge detection, and sharpening. Let us assume an input image frame of dimensions Win x Hin. The convolutional kernel with dimensions K x K is considered for convolution with a stride of S and P padding. The size of the output (Wout x Hout) of convolution layer is given by equations 2 and 3.

$$W_{out} = (W_{in}\text{-}K+2P)/S+1 \quad (2)$$

$$H_{out} = (H_{in}\text{-}K+2P)/S+1 \quad (3)$$

There are three convolutional layers in the CNN model's design. The following standard equation in (4) is typically used to represent the convolutional layer output.

$$y_j^n = f\left(\sum_{i \in c_j} y_i^{n-1} * k_{ij}^n + \zeta_j^n\right) \quad (4)$$

Where n is the nth layer, kij represents the convolutional kernel, ζj is the bias and the input maps are symbolized by cj. The CNN employs the piecewise linear ReLU activation function, which yields zero in the absence of a positive input; otherwise, it outputs the input directly. It is formulated as shown in equation (5).

$$f(x) = \max(0,x) \quad (5)$$

3) Pooling Layer: A convoluted image can be too large and therefore must be shrunk without sacrificing any of its feature. Max and min pooling are two varieties of pooling. The proposed CNN makes use of max pooling. The maximum value from the specific region is chosen in the max pooling layer. Each convolutional layer uses a max-pooling layer having a 2x2 kernel size to downsample the output by a factor of 2. The formula for calculating the output max pooling spatial shape as shown in equation (6)

$$\lfloor (Ix\text{-}P)/S \rfloor + 1 \quad (6)$$

Where P is the pooling window size, S presents the amount of strides, and Ix represents the input x or y shape.

4) Flattening Layer: In order to feed a multi-dimensional matrix into the fully connected layer, this layer converts it to a 1-dimensional array.

5) Fully Connected or Dense Layers: The model has two fully connected layers with 256 and 128 units, respectively. The dense layer of a CNN is a highly coupled layer that provides learning features based on all potential combinations of the features from the preceding layer. It carries out using equation (7):

$$\text{Output} = \text{activation}(\text{dot}(\text{input},\text{kernel}) + \text{bias}) \quad (7)$$

6) Output Layer: The output layer of the model comprises of 24 units, which are activated by a Softmax activation function. The Softmax function generates a probability distribution over the 24 possible classes to which the input image can belong. In classification models, it is common to utilize the softmax activation function in the output layer. This function helps convert the output of the last layer into a probability distribution across the various classes. The output is transformed into a vector of values ranging from 0 to 1. The sum of all the values in the vector is equal to 1, with each value denoting the likelihood or probability of the input image being associated with a specific class.. The model predicts the class with the highest probability as the output. The Softmax activation function can be mathematically expressed as shown in equation 8.

$$f(p) = softmax = \sigma(p^{\rightarrow}) = \frac{e^{pi}}{\sum_{j=1}^{k} e^{pj}} \quad (8)$$

D. Model Training

The model underwent training on the training dataset for a duration of 5 epochs with a batch size of 64. This number of epochs was chosen because the dataset was relatively small, so the smaller number of epochs was sufficient to achieve reasonable performance while minimizing the training time. The decision was also made to prevent overfitting to the training data which can be caused by prolonged training, resulting in poor performance on new and previously unseen data. The validation dataset was used to obtain optimal settings for the CNN hyper-parameters.

E. Model Evaluation

To assess the model's performance on novel, unseen data, it was evaluated using the test dataset. Various metrics, including accuracy, precision, recall, and F1-score, were computed based on the test data to gauge the model's effectiveness.

F. Model Deployment

The model was deployed for testing in real time. The application was able to access the device camera to acquire live video feed, and then display the model's prediction on the screen. In order for the system to detect signs, the user background should preferably be clutterless and there needs to be sufficient lighting. Once the hand is detected, the application will generate the equivalent text representation of the sign.

IV. Experiments and Results

The overall training accuracy was determined to be 99.86% while validation accuracy was determined to be 99.94%. The test accuracy was determined to be 94.68%. Figures 5 exhibits the curves for training and validation accuracy and Fig 6. depicts the training and validation loss over 5 epochs. A confusion matrix and the calculated test accuracy, precision, recall, f1- score were obtained based on the test dataset using scikit learn library in Python.

Accuracy is the most effective performance metric and is computed as the ratio of correct predictions to all predictions. The mathematical expression is given in equation (9):

$$Accuracy = \frac{number\ of\ correct\ precictions}{number\ of\ all\ predictions} = \frac{TP+TN}{TP+FP+FN+TN} \quad (9)$$

The degree of precision is determined by the proportion of accurately predicted positives to all predicted positive observations. The mathematical expression is given in equation (10):

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

The ratio of correctly predicted positive observations to the total number of actual positive observations is known as

recall. The mathematical expression is given in equation (11):

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

These are the key words used in the equations for the evaluation metrics:

True Positives (TP) are correctly anticipated positive values, demonstrating that both the actual class value and the projected class value are accurate.

True Negatives (TN) are properly predicted negatives, or wrong values, which show that there is a discrepancy between the actual class value and the anticipated class value.

False Positives (FP) are instances where the actual class is incorrect despite the predicted class being correct.

False Negatives (FN) are instances in which the actual class is true, but the predicted class is false.

Table 1 depicts the recall and f1-score of recognition of each letter class with Class labels 0 to 23 representing letters A to Y excluding J. These metrics were calculated based on the test data.

TABLE I. MODEL PRECISION, RECALL AND F1 SCORE

Class	*precision*	*recall*	*f1-score*
0	1.00	0.98	0.99
1	0.89	0.99	0.93
2	1.00	0.98	0.99
3	0.73	0.80	0.77
4	0.95	0.99	0.97
5	0.99	0.99	0.99
6	1.00	1.00	1.00
7	0.99	1.00	0.99
8	1.00	0.67	0.80
9	0.86	0.98	0.92
10	0.96	1.00	0.98
11	0.93	0.87	0.90
12	0.88	1.00	0.94
13	1.00	1.00	1.00
14	1.00	1.00	1.00
15	1.00	1.00	1.00
16	0.83	0.97	0.89
17	0.97	0.90	0.94
18	1.00	1.00	1.00
19	0.92	0.96	0.94
20	0.94	0.88	0.91
21	1.00	0.78	0.87
22	1.00	1.00	1.00
23	1.00	1.00	1.00

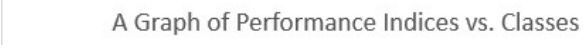
Fig. 4. Graph of precision, recall and f1-score metrics calculated using the test dataset

The precision, recall, and F1-score metrics are plotted on a graph as a line chart as shown in Fig. 4. The x-axis represents the 24 different classes, and the y-axis represents the corresponding metric scores. The lines are color-coded for ease of interpretation. From the graph, all the classes have relatively high precision, recall, and F1-score. These are indicative of good prediction performance.

The curve in Fig. 5 shows the accuracy of the model on a training dataset and the validation dataset for 5 epochs or training iterations. Fig. 6 shows the loss of the model on both the training and validation dataset for 5 epochs or training iterations. The x-axis in both figures represent the number of training iterations or epochs, while the y-axis represents the accuracy of the model in Fig. 5 and the loss in Fig. 6 on the corresponding dataset.

In Fig. 5, at the beginning of training, the training accuracy is lower. This is because the CNN model has not yet learned to recognize hand gestures and is essentially making random guesses. At the start of training, the model's weights are randomly initialized, resulting in a lower training accuracy. As the training progresses, the model updates its weights, which leads to an increase in accuracy on both the training and validation sets. As the number of epochs increases, the training accuracy gradually improves, as the model learns to recognize hand gestures by adjusting the weights of its neurons. The validation accuracy remained consistently high for all epochs with a slight decrease between the first and third epoch. This means that the CNN model was not overfitting to the training data, which is important because overfitting can cause the model to perform poorly on new data.

In Fig. 6, it is shown that at the beginning of training, the training loss is high. This is because the CNN model is making random guesses and has not yet learned to recognize hand gestures accurately. The loss curve represents how the



model's weights affect its ability to minimize the difference between the predicted output and the actual output. At the start of training, the model's loss is high because its weights are random, and it is essentially making random guesses. As the training progresses, the model updates its weights, which leads to a decrease in the loss on both the training and validation sets. As the number of epochs increases, the training loss gradually decreases, indicating that the CNN model is getting better at recognizing hand gestures on the training dataset. The validation loss remained consistently near 0 for all epochs with a slight increase between the first and third epoch. This would suggest that the CNN model is able to recognize hand gestures accurately with a high level of confidence. This would provide a strong indication that the CNN model is a good candidate for use in sign language prediction, as it is able to make accurate predictions with a very low level of error.

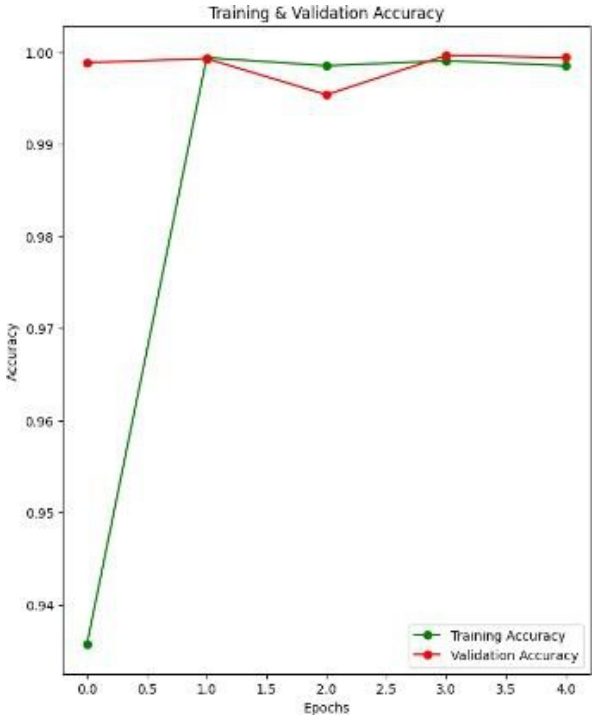


Fig. 5. Graph of training and validation accuracy for 5 epochs

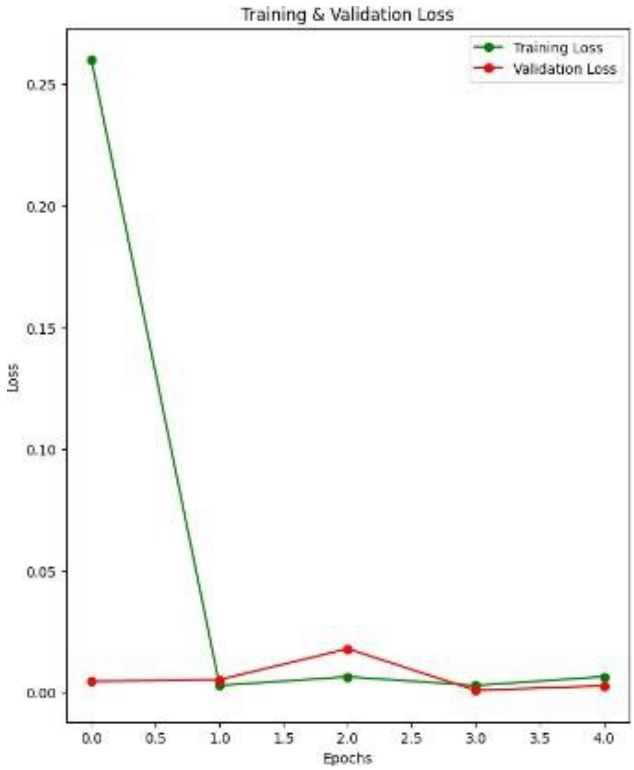


Fig. 6. Graph of training and validation loss for 5 epochs

Figure 7 displays the confusion matrix of the model, which was generated using the test dataset. It is a 24x24 table that provides an overview of the multiclass classification model's performance by comparing the predicted and actual labels for each class. To verify the model's accuracy on the test dataset, one can sum the number of true positives and divide it by the total number of instances, as expressed in equation (12). This equation offers a reliable method to assess the accuracy of the model using the information provided in the confusion matrix.

$$Accuracy = \frac{total\ number\ of\ true\ positives}{total\ number\ of\ instances} \quad (12)$$

Therefore, the test accuracy of the model can be calculated as seen in equation (13):

$$Accuracy = \frac{\begin{array}{c}600 + 608 + 603 + 499 + 621 + 611 + 611 \\ +619 + 422 + 624 + 635 + 555 + 618 + 620 \\ +630 + 625 + 629 + 569 + 621 + 605 + 557 \\ +476 + 614 + 610\end{array}}{14979} \quad (13)$$

Which would yield an accuracy value of 94.68%.

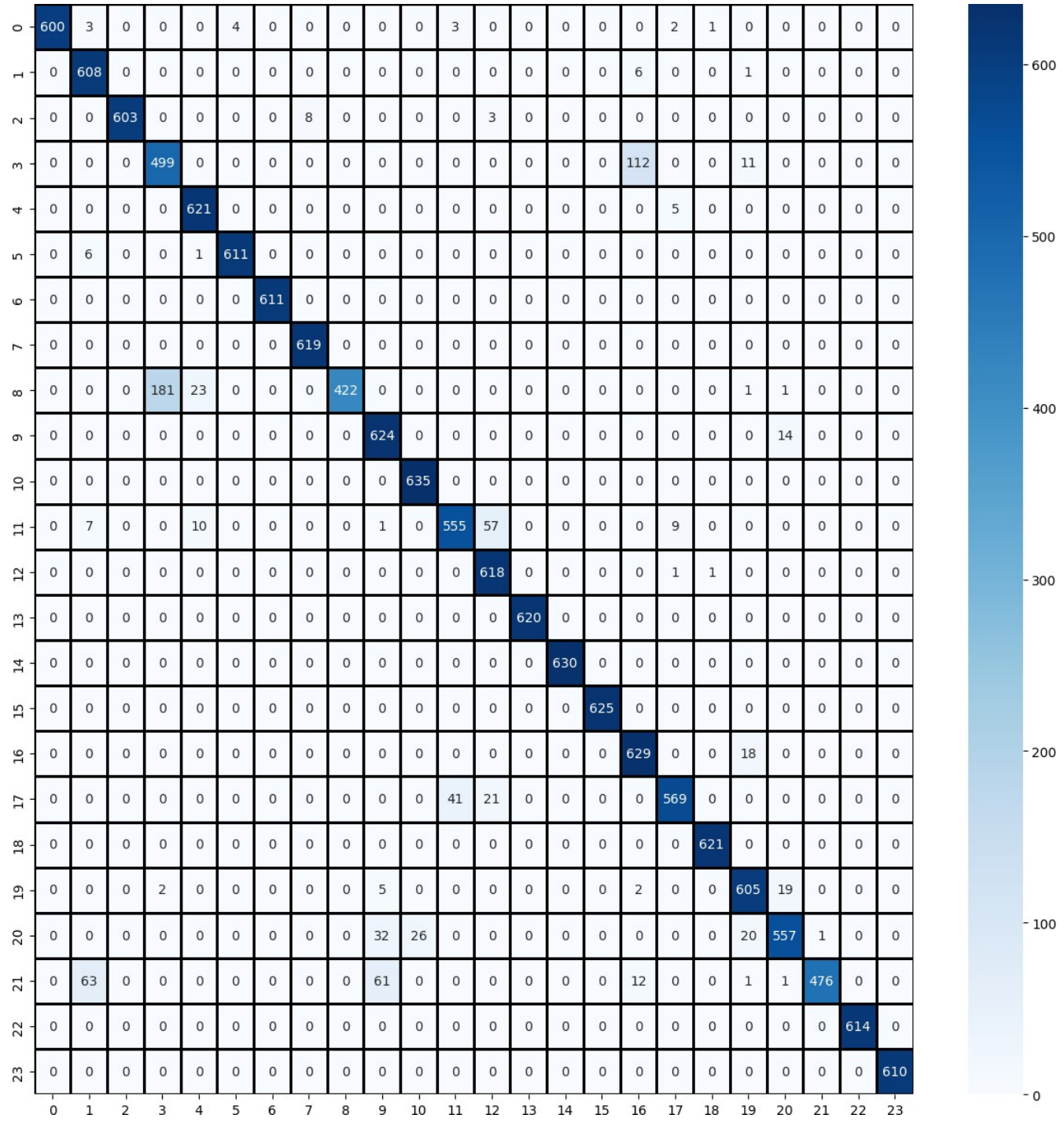


Fig. 7. Confusion matrix of the CNN model

V. CONCLUSION

This system was successfully developed, implemented, tested, and deployed. The paper was able to emphasize the significance of sign language recognition systems in the lives of the deaf and the hearing-impaired. It was also able to recognize sign language gestures with high accuracy. The system was able to recognize gestures in real-time. However, additional data in different environmental conditions may improve the model's efficacy and reliability. Also, the system's recognition capabilities are currently limited to static ASL alphabet gestures. It does not consider the dynamic and continuous nature of sign language, which involves intricate movements, facial expressions, and body language. Future research should aim to expand the system to recognize dynamic sign language gestures for more comprehensive communication. The system has also been specifically trained and tested on the ASL alphabet. However, sign languages vary across different regions and countries, each with their own unique vocabulary and grammar. The system's performance on other sign languages remains unknown and requires further investigation and adaptation to ensure its generalizability.

ACKNOWLEDGMENT

The authors acknowledge the financial support offered by Covenant University in the publication of this paper.

REFERENCES

[1] S. A. Khan, "Importance of hearing and hearing loss treatment & recovery," *IJSA*, vol. 3, no. 1, pp. 14–16, 2022.

[2] K. Aashritha and V. M. Manikandan, "Assistive Technology for Blind and Deaf People: A Case Study," in *Machine Vision and Augmented Intelligence: Select Proceedings of MAI 2022*, Springer, 2023, pp. 539–551.

[3] S. Treat, "Deaf Education: Gallaudet university: How deaf education and special education is being advanced in Nigeria," *Retrieved Nov*, vol. 2, p. 2016, 2016.

[4] C. J. Eleweke, I. O. Agboola, and S. I. Guteng, "Reviewing the pioneering roles of Gallaudet University alumni in advancing deaf education and services in developing countries: Insights and challenges from Nigeria," *Am. Ann. Deaf*, vol. 160, no. 2, pp. 75–83, 2015.

[5] E. Sammari and A. Naceur, "The Effectiveness of an Intervention Program for the Development of Social and Emotional Capacities of Children with a Hearing impairment," *Psychology*, vol. 13, no. 7, pp. 1025–1062, 2022.

[6] S. Shave, C. Botti, and K. Kwong, "Congenital sensorineural hearing loss," *Pediatr. Clin.*, vol. 69, no. 2, pp. 221–234, 2022.

[7] L. Lee, "The Importance of Learning Deaf Culture through a Black Deaf Perspective in the Field of Communication Sciences and Disorders," 2022.

[8] S. Colibaba, I. Gheorghiu, A. Colibaba, O. Ursa, C. Antoniţă, and R. Cîrşmari, "The Voice Project: Habilitating Hearing-Impaired Children to Recover Hearing and Lead a Normal Life," in *2022 E-Health and Bioengineering Conference (EHB)*, IEEE, 2022, pp. 1–4.

[9] Y. Xusnora and A. Yulduz, "Ways to develop vocabulary in children with hearing impairment," in *E Conference Zone*, 2022, pp. 229–230.

[10] K. N. Robillard, E. de Vrieze, E. van Wijk, and J. J. Lentz, "Altering gene expression using antisense oligonucleotide therapy for hearing loss," *Hear. Res.*, p. 108523, 2022.

[11] A. S. Dhanjal and W. Singh, "An optimized machine translation technique for multi-lingual speech to sign language notation," *Multimed. Tools Appl.*, vol. 81, no. 17, pp. 24099–24117, 2022.

[12] S. H. Hamerdinger and C. J. Crump, "Sign language interpreters and clinicians working together in mental health settings," *Routledge Handb. Sign Lang. Transl. Interpret.*, 2022.

[13] A. H. Alhafdee, H. Abbas, and H. I. Shahadi, "Sign Language Recognition and Hand Gestures Review," *Kerbala J. Eng. Sci.*, vol. 2, no. 4, pp. 192–316, 2022.

[14] W. Lu *et al.*, "Artificial Intelligence–Enabled Gesture-Language-Recognition Feedback System Using Strain-Sensor-Arrays-Based Smart Glove," *Adv. Intell. Syst.*, p. 2200453, 2023.

[15] D. W. Alausa *et al.*, "PalmMatchDB: An On-Device Contactless Palmprint Recognition Corpus," in *2023 IEEE 3rd International Conference on Power, Electronics and Computer Applications, ICPECA 2023*, IEEE, 2023, pp. 318–325. doi: 10.1109/ICPECA56706.2023.10076097.

[16] E. Hisham and S. N. Saleh, "ESMAANI: A Static and Dynamic Arabic Sign Language Recognition System Based on Machine and Deep Learning Models," in *2022 5th International Conference on Communications, Signal Processing, and their Applications (ICCSPA)*, IEEE, 2022, pp. 1–6.

[17] A. Ademola, T. E. Somefun, A. F. Agbetuyi, and A. Olufayo, "Web based fingerprint roll call attendance management system," *Int. J. Electr. Comput. Eng.*, vol. 9, no. 5, pp. 4364–4371, 2019, doi: 10.11591/ijece.v9i5.pp4364-4371.

[18] J. Eunice, Y. Sei, and D. J. Hemanth, "Sign2Pose: APose-Based Approach for Gloss Prediction Using a Transformer Model," *Sensors*, vol. 23, no. 5, p. 2853, 2023.

[19] T. A. Siby, S. Pal, J. Arlina, and S. Nagaraju, "Gesture based Real-Time Sign Language Recognition System," in *2022 International Conference on Connected Systems & Intelligence (CSI)*, IEEE, 2022, pp. 1–6.

[20] E. Adetiba *et al.*, "FEDGEN Testbed: A Federated Genomics Private Cloud Infrastructure for Precision Medicine and Artificial Intelligence Research," in *Communications in Computer and Information Science*, Springer, 2022, pp. 78–91. doi: 10.1007/978-3-030-95630-1_6.

[21] A. M. Soliman, M. M. Khattab, and A. M. Ahmed, "Arabic Sign Language Recognition System: Using an Image-Based hand Gesture Detection Method to help Deaf and Dump Children to Engage in Education,"vol. 32, no. 58, pp. 1–28, 2023.

[22] K. Amrutha and P. Prabu, "ML based sign language recognition system," *2021 Int. Conf. Innov. Trends Inf. Technol. ICITIIT 2021*, 2021, doi: 10.1109/ICITIIT51526.2021.9399594.

[23] P. Pranav and R. Katarya, "Optimal Sign language recognition employing multi-layer CNN," in *2022 4th International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, IEEE, 2022, pp. 288–293.

[24] Y. Farhan, A. A. Madi, A. Ryahi, and F. Derwich, "American Sign Language: Detection and Automatic Text Generation," in *2022 2nd International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*, IEEE, 2022, pp. 1–6.

[25] A. Thakur, P. Budhathoki, S. Upreti, S. Shrestha, and S. Shakya, "Real Time Sign Language Recognition and Speech Generation," *J. Innov. Image Process.*, vol. 2, no. 2, pp. 65–76, 2020, doi: 10.36548/jiip.2020.2.001.

[26] S. Shrenika and M. Madhu Bala, "Sign Language Recognition Using Template Matching Technique," *2020 Int. Conf. Comput. Sci. Eng. Appl. ICCSEA 2020*, pp. 5–9, 2020, doi: 10.1109/ICCSEA49143.2020.9132899.

[27] L. K. S. Tolentino, R. O. Serfa Juan, A. C. Thio-ac, M. A. B. Pamahoy, J. R. R. Forteza, and X. J. O. Garcia, "Static sign language recognition using deep learning," *Int. J. Mach. Learn. Comput.*, vol. 9, no. 6, pp. 821–827, 2019, doi: 10.18178/ijmlc.2019.9.6.879.

[28] S. Puri, M. Sinha, S. Golaya, and A. K. Dubey, "Indian sign language recognition using python," in *Emerging Technologies in Data Mining and Information Security*, Springer, 2021, pp. 427–434.

[29] S. Sharma and K. Kumar, "ASL-3DCNN: American sign language recognition technique using 3-D convolutional neural networks," *Multimed. Tools Appl.*, vol. 80, no. 17, pp. 26319–26331, 2021, doi: 10.1007/s11042-021-10768-5.

[30] M. Quinn and J. I. Olszewska, "British sign language recognition in the wild based on multi-class SVM," in *2019 federated conference on computer science and information systems (FedCSIS)*, IEEE, 2019, pp. 81–86.