Character Recognition of Auslan Sign Language using Neural Network

Mateen Mureed[1] ,Mir Atif[1] ,Fahad Ali Abbasi[2]
[1]Sindh Madressatul Islam University/Department of Artificial Intelligence, Karachi, Pakistan
[2]Bank Alhabib Limited IT Division

E-mail: mateenmurid@gmail.com, atifmeer78@gmail.com, fahad.ali.abbasi@gmail.com.

Corresponding Author: Mateen Mureed, mateenmurid@gmail.com

Received: 11-03-2023; **Accepted:** 27-05-2023; **Published**: 01-07-2023

Abstract: People which have inability to speak use different modes of communication to convey their message the most common one is sign language they use hand gestures and orientation of arms with facial expressions. To make it easy for the normal people to communicate with deaf people we use neural networks to build an AI model application. We use Densely Connected Convolutional Neural Networks to capture the features of image dataset with label of sign languages. The goal of this work is to take the first crucial step in using sign language to overcome the communication gap between hearing people and deaf and people. For this we use image dataset consists of 1300 images. By using Convolutional Neural Network, we got an accuracy of 92%.

Keywords: Sign Language, Machine Learning, Deep Learning

1. Introduction

Over 70 million individuals worldwide are deaf, according to the World Federation of the Deaf. Over 80% of them reside in underdeveloped nations. They employ more than 300 distinct sign languages collectively. The sharing of ideas, feelings, and information is made possible via communication, which is a key component of human contact. However, traditional spoken languages might not be available to those who are deaf or hard of hearing. In these situations, sign languages are used as the main form of communication. To transmit meaning and improve communication, sign languages combine hand gestures, facial expressions, lip motions, and body language [13]. The gestures and syntax used in each of these languages differ depending on the locale.

For the deaf community, sign language makes use of hand movements, facial expressions, lip patterns, and orientation and movement of the hands, arms, or body [1]. Regional sign languages include British Sign Language (BSL), Indian Sign Language (ISL), and others. Convolutional neural networks (CNN's), in particular, have made significant strides in deep learning algorithms in recent years, revolutionizing many industries, including image identification and computer vision. Researchers have started looking into the possibilities of these algorithms for use in sign language recognition. It is now possible to create automated systems that can decipher and comprehend sign language motions using CNN's and other deep learning techniques, facilitating communication between the hearing and deaf communities [12]. We provide a strategy for training an Auslan sign language recognition system that combines image data pre-processing approaches and CNN-based feature extraction. To identify and arrange our 36 directories of Auslan sign language image files, we use the Image Data Generator library. We attain an excellent accuracy rate of 92% via numerous training phases, and a validation accuracy of 95%.

Our initiative is significant because it has the potential to significantly improve accessibility [2] and communication for the deaf community. By removing barriers and promoting inclusion, a reliable and accurate sign language recognition system can help deaf people communicate more naturally with the hearing population. Additionally, by applying deep learning algorithms, particularly CNN's [4], and investigating transfer learning strategies to get around the problem of little training data, our research advances the field of sign language recognition. The steps involved in data preprocessing, the structure of our CNN model, and the training procedure will all be covered in more detail in the sections that follow. We will also discuss the outcomes of our research, emphasizing the effectiveness and precision of our sign language recognition system. We hope that this research will significantly increase sign language detection and its use to help the deaf community communicate more effectively.

2. Literature Review

Due to developments in deep learning techniques and their use in computer vision problems, sign language recognition has attracted a lot of attention lately [6]. The most recent research articles on various methods and strategies for sign language recognition are reviewed in this area. A thorough analysis of the literature on deep learning's use in the recognition of sign language was carried out by Hanke, Hirschfeld, and Herrmann in 2021[11]. They examined a wide range of papers to find common approaches and prevailing themes. The review underlined the value of large-scale sign language datasets for building reliable models and highlighted the efficacy of Convolutional Neural Networks (CNNs) for sign language recognition tasks. With the dearth of sign language datasets, transfer learning has become a potent tool. A survey of deep learning-based American Sign Language (ASL) identification was presented by Pu et al. in 2020 [12]. They talked about how to modify models trained on massive datasets to recognize ASL signals using pre-trained models and transfer learning techniques. According to the poll, transfer learning is excellent in increasing recognition accuracy and minimizing the requirement for substantial training data.

Additionally, Zhang and Gong (2020) [13] carried out a thorough analysis of sign language recognition, offering insights into different approaches and developments in the field. The usage of CNNs, recurrent neural networks, and attention mechanisms for sign language recognition were also included in the review. It also emphasized recent advancements in multimodal techniques that blend skeletal and visual data to enhance recognition precision. Dreuw and Deselaers (2017) [14] suggested a method based on recurrent neural networks (RNNs) for continuous sign language recognition. They investigated how to store temporal relationships in sign language sequences using long short-term memory (LSTM) units. The work demonstrated the potential of RNNs for sequential data handling and continuous sign language recognition. Athanasios and Dimitrios (2020) [15] published a review that was specifically concerned with the use of CNNs for the recognition of sign language. They talked about the efficiency of various CNN architectures in identifying features in photographs of sign language. The review also stressed how crucial data augmentation methods are for boosting CNN models' capacity for generalization.

Benjamin Schrauwen, Sander Dieleman, PieterJan Kindermans, and Lionel Pigou [1]. Convolutional neural networks (CNNs), GPU acceleration, and Microsoft Kinect are all used in their contribution as recognition systems. CNNs can automate the feature construction process so that intricate, handcrafted features are not required. They had a high degree of accuracy while identifying 20 Italian gestures. Their predictive model had a cross-validation accuracy of 91.7% and could generalize training. In order to recognize Portuguese sign language, Paulo Trigueiros, Fernando Ribeiro, and Lus Paulo Reis [9] have proposed a real-time vision-based system. They extracted hand features using the Kinect Camera. Open source Dlib library, a general-purpose cross-platform C++ library capable of SVM multiclass classification, was utilized for model training and gesture categorization.

Neha V. Tavan and Professor A.V. Deorankar [7] In their research created an algorithm to extract HOG characteristics. The artificial neural network that was later employed for gesture recognition was trained using these features. Using form analysis, Haitham Hasan and S. Abdul-Kareem [8] have presented a method for recognizing hand gestures. They classified six static hand gestures—open, close, cut, paste, maximize, and minimize—using a neural network-based method. They employed a special multi-layer sensory neural network with a backpropagation learning algorithm for categorization. The accuracy they were able to obtain was 86.38 percent. S. Nagarajan and S. Nagarajan [9] employed multiclass SVM and Edge Oriented Histogram features. In order to classify the input sign language alphabets, the edge histogram count is retrieved as a feature and applied to a multiclass SVM. The system's average accuracy was 93.75 percent. Pushkar Shukla, Bhumika Gupta, and Ankush Mittal [10] To extract features for a picture, they have employed HOG and SIFT. Then, a single matrix made up of all of these traits is created. These matrices' correlation is calculated and supplied to a K-Nearest Neighbor Classifier. 179 out of 200 gestures could be successfully detected.

3. Convolutional Neural Network (CNN)

In order to detect and extract features from image-based data, such as sign language videos or photos, convolutional neural networks are frequently employed in sign language recognition. The objective of understanding sign language is to recognize the hand gestures, movements, and forms that are used to represent words and phrases. Because they can examine patterns and features at several levels of abstraction, from straightforward edges and forms to more intricate movements and gestures, CNNs are particularly good at this task. Even in noisy and complicated contexts, most of the people use to train models to understand sign language with great accuracy by employing CNNs [3]. These models can be used to create sign language translation systems, which will improve communication for people who are hard of hearing or deaf. In general, the application of CNNs in sign language recognition has helped to increase the precision and dependability of these systems, increasing their usability and accessibility for people who communicate using sign language. We used 36 classes 26 for alphabets and 9 classes for digits for image classification CNN models pass through different parts of the Sequential layer, dense layer, maxpooling2d, and conv2d layers [3,5].

4. Dataset

The dataset we used in our project is the Auslan Sign Language (fingerspelling) dataset downloaded from Kaggle which is the huge online community for data scientists and machine learning experts and contains thousands of published datasets [10]. The dataset contains more than 71 thousand images for the Auslan Fingerspelling Hand Signs with a Gaussian Blur Filter applied on all the images so that images can be used for better feature extraction. Each class has around twenty thousand images of multiple poses. There are a total of 36 classes in the dataset. the dataset can be downloaded from this site. (AUSLAN SIGN LANGUAGE (FINGERSPELLING) DATASET | Kaggle).

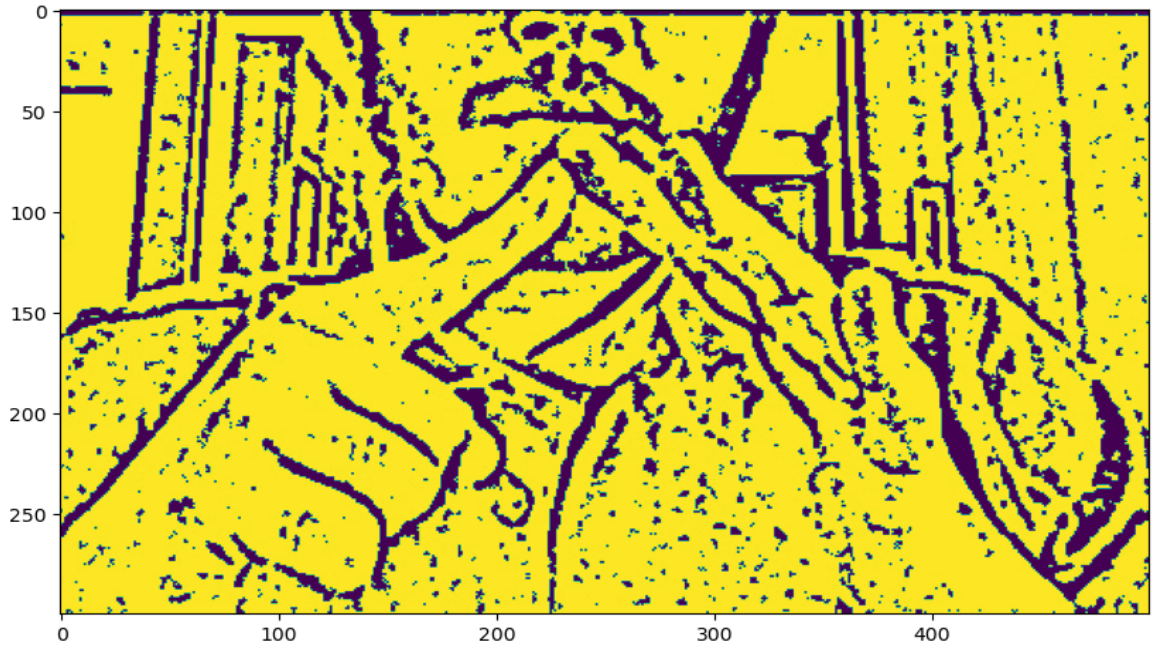

Figure. 1 Gaussian Filter image of I Character

Figure 1. shows an image of an I character sign applied by Gaussian Filter to it. We applied the Gaussian filter so that our model can learn better and only focus on the main features of hand gestures. This will help in improving the accuracy of our model.

5. Methodology

Figure 2. Shows the process of dataset generation process by using Image Data Generator. For this, we used Image Data Generator which is a class in Keras that generates batches of image data with real-time data augmentation. Then we rescale the pixels of images between 0 and 1. Applies zoom to the images in the range of 0.8 to 1.2 and shifts the images vertically by a maximum of 20% of the height. After that, we resize the images in 150 X 150 pixels with a batch size of 36 that applies the class mode to categorical for the training and validation of our model.

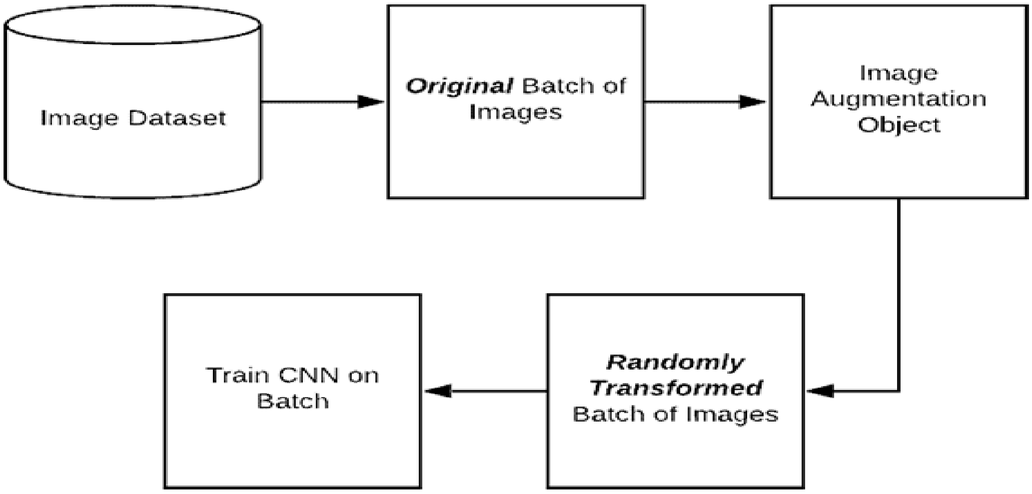

Figure. 2 Data Generation Process

5.1 Model Training

```
 Layer (type)                    Output Shape              Param #
=================================================================
 conv2d (Conv2D)                 (None, 148, 148, 32)      896

 max_pooling2d (MaxPooling2D      (None, 74, 74, 32)       0
 )

 conv2d_1 (Conv2D)               (None, 72, 72, 64)        18496

 max_pooling2d_1 (MaxPooling      (None, 36, 36, 64)       0
 2D)

 conv2d_2 (Conv2D)               (None, 34, 34, 128)       73856

 max_pooling2d_2 (MaxPooling      (None, 17, 17, 128)      0
 2D)

 conv2d_3 (Conv2D)               (None, 15, 15, 128)       147584

 max_pooling2d_3 (MaxPooling      (None, 7, 7, 128)        0
 2D)

 flatten (Flatten)               (None, 6272)              0
...
Total params: 6,701,284
Trainable params: 6,701,284
Non-trainable params: 0
```

Figure. 3 CNN model for image classification

Figure3. shows the CNN model for the sign language image classification with 36 classes having 26 alphabets and 9 digits classes.

- A model starts with a Sequential model which is a linear stack of layers that passes with a convolutional layer with 32 filters of size 3x3, using the ReLU activation function.

- The shape of the input images is 150,150,3 which means that the color of the images is RGB and pixels are 150 X 150.

- MaxPooling2 adds a max pooling layer with a pool size of 2 x 2 which is used for reducing the spatial dimensions of the output of the previous convolutional layer by a factor of two.
- Conv2D (64, (3, 3), activation='relu' adds another convolutional layer with 64 filters of size 3x3, using the ReLU activation function. Followed by a max pooling layer with a pool size of 2x2.
- Then it adds another two convolutional layers of Conv2D (128, (3, 3) with 128 64 filters of size 3x3, using the ReLU activation function. Followed by a max pooling layer with a pool size of 2x2.
- After that, we add a flatten which flattens the output of the previous layer into a 1D vector.
- Finally, we add two fully connected layers the first layer adds 1024 neurons and a Relu activation function and the second fully connected layer adds 36 classes of neurons with a sigmoid activation function.

Overall, it has four convolutional layers followed by a pooling layer then flattens the outputs of the previous layer and in the last adds two fully connected dense layers with relu and sigmoid activation functions. The number of total parameters is around 6.7 million. We trained our model using the above-specified parameters that contain information about the training and validation performance of the model of each epoch for this we used 20 epochs and 200 validation steps.

```
Epoch 1/20
200/200 [==============================] - 136s 628ms/step - loss: 3.5790 - accuracy: 0.0318 - val_loss: 4.4366 - val_accuracy: 0.0274
Epoch 2/20
200/200 [==============================] - 118s 593ms/step - loss: 3.2625 - accuracy: 0.1033 - val_loss: 2.9020 - val_accuracy: 0.1757
Epoch 3/20
200/200 [==============================] - 115s 576ms/step - loss: 2.5340 - accuracy: 0.2746 - val_loss: 1.6310 - val_accuracy: 0.4919
Epoch 4/20
200/200 [==============================] - 111s 557ms/step - loss: 1.8767 - accuracy: 0.4520 - val_loss: 1.6563 - val_accuracy: 0.4807
Epoch 5/20
200/200 [==============================] - 106s 530ms/step - loss: 1.3500 - accuracy: 0.5794 - val_loss: 1.0006 - val_accuracy: 0.6881
Epoch 6/20
200/200 [==============================] - 105s 527ms/step - loss: 1.0370 - accuracy: 0.6786 - val_loss: 0.7716 - val_accuracy: 0.7493
Epoch 7/20
200/200 [==============================] - 111s 554ms/step - loss: 0.8531 - accuracy: 0.7307 - val_loss: 0.6530 - val_accuracy: 0.7972
Epoch 8/20
200/200 [==============================] - 99s 498ms/step - loss: 0.7195 - accuracy: 0.7732 - val_loss: 0.5946 - val_accuracy: 0.8012
Epoch 9/20
200/200 [==============================] - 108s 540ms/step - loss: 0.5966 - accuracy: 0.8069 - val_loss: 0.3354 - val_accuracy: 0.8860
Epoch 10/20
200/200 [==============================] - 108s 540ms/step - loss: 0.5453 - accuracy: 0.8253 - val_loss: 0.3443 - val_accuracy: 0.8835
Epoch 11/20
200/200 [==============================] - 107s 536ms/step - loss: 0.4794 - accuracy: 0.8511 - val_loss: 0.3253 - val_accuracy: 0.8839
Epoch 12/20
200/200 [==============================] - 107s 536ms/step - loss: 0.4081 - accuracy: 0.8685 - val_loss: 0.3862 - val_accuracy: 0.8861
Epoch 13/20
...
Epoch 19/20
200/200 [==============================] - 104s 519ms/step - loss: 0.2562 - accuracy: 0.9185 - val_loss: 0.2228 - val_accuracy: 0.9250
Epoch 20/20
200/200 [==============================] - 103s 516ms/step - loss: 0.2295 - accuracy: 0.9262 - val_loss: 0.1467 - val_accuracy: 0.9535
```

Figure. 4 Visualization of Training Progress

Figure. 4 shows that we got 92% accuracy from training our dataset by using a CNN model and 95% of validation accuracy.

5.2 Results & Accuracy:

Model	Accuracy	Validation Accuracy
CNN	92%	95%

Table. 1 Accuracy Result

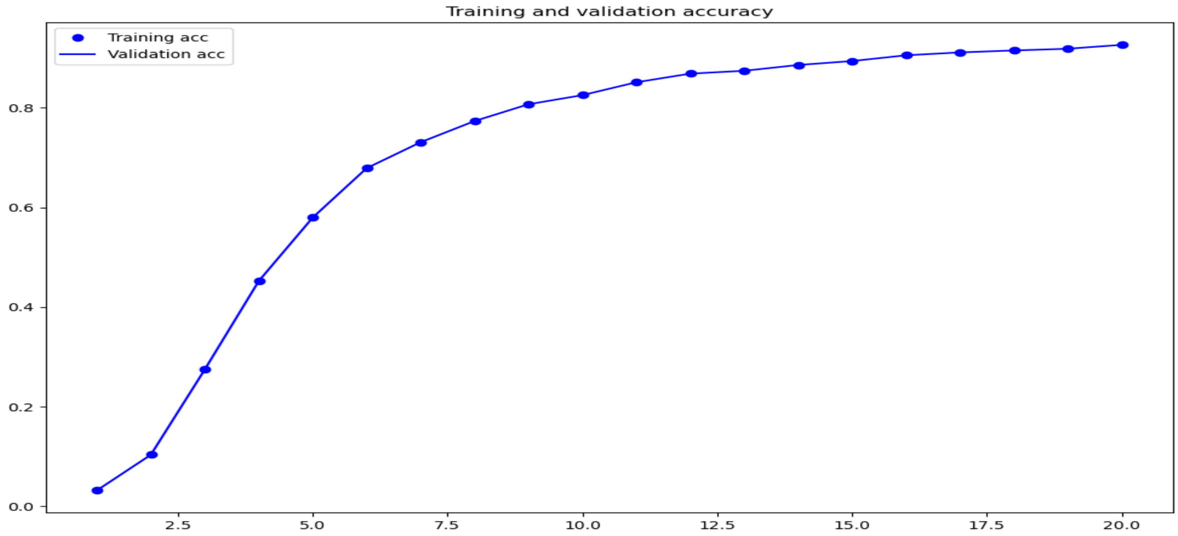

Figure. 5 Training n Validation Accuracy

Fig. 5 indicates the progress of the training model which starts with 30% accuracy and ends with 92% accuracy of the model and 95% of validation accuracy at 20 epochs. At each iteration model increases the accuracy. We only used 20 epochs because of fewer resources which may increase the accuracy if we pass around 100 epochs it may get up to 95$ or more.

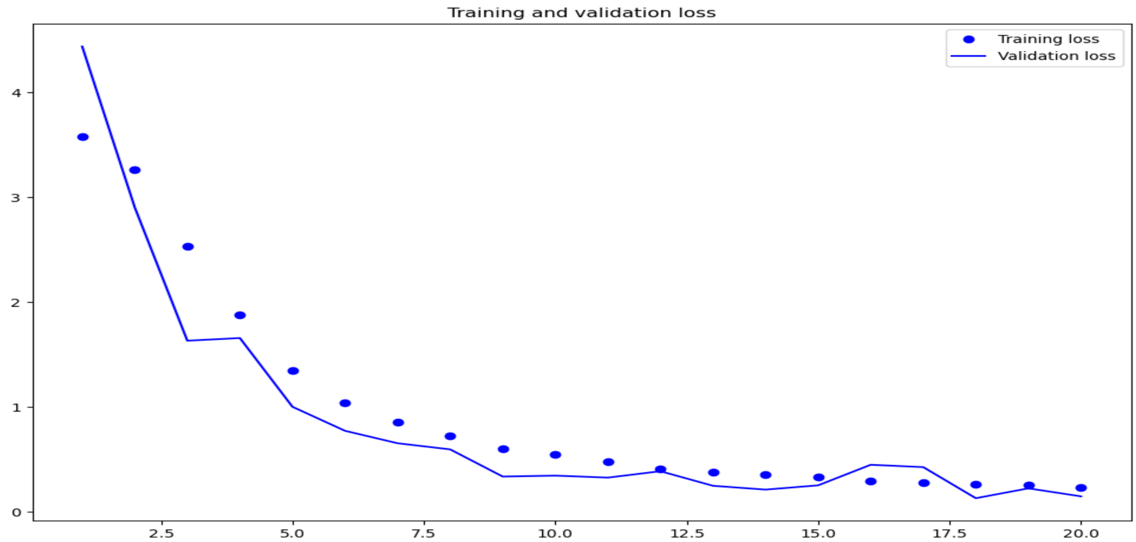

Figure. 6 Training and Validation Loss

6. Real time results

After successfully training the model, we taken some real time tests on computer vision a well-known popular python library OpenCV. We saved our trained model in model.h5 format and then implemented on OpenCV to test on real time recognition some of the results are shown below in Fig. 7,8,9.

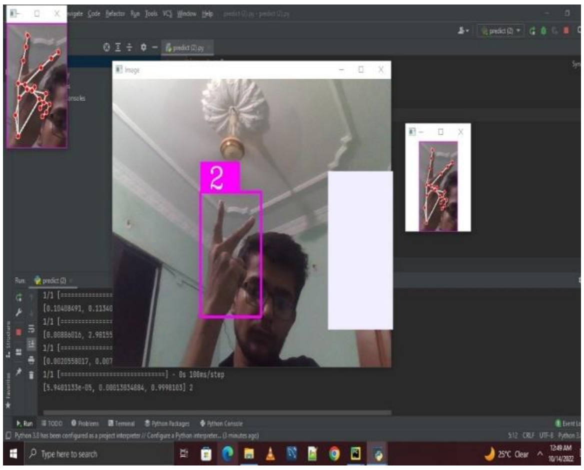

Figure. 7 Character 2 Recognition

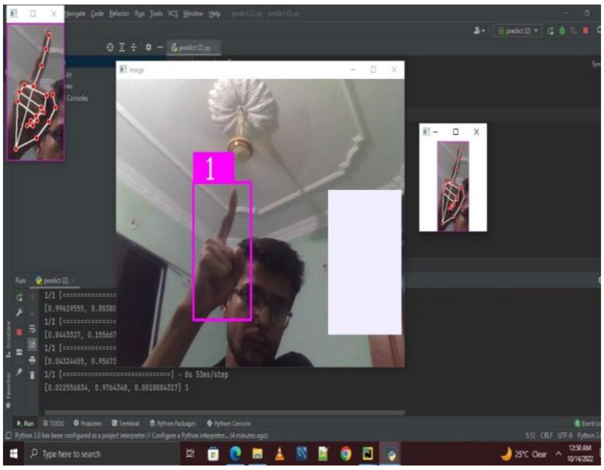

Figure. 8 Character 1 Recognition

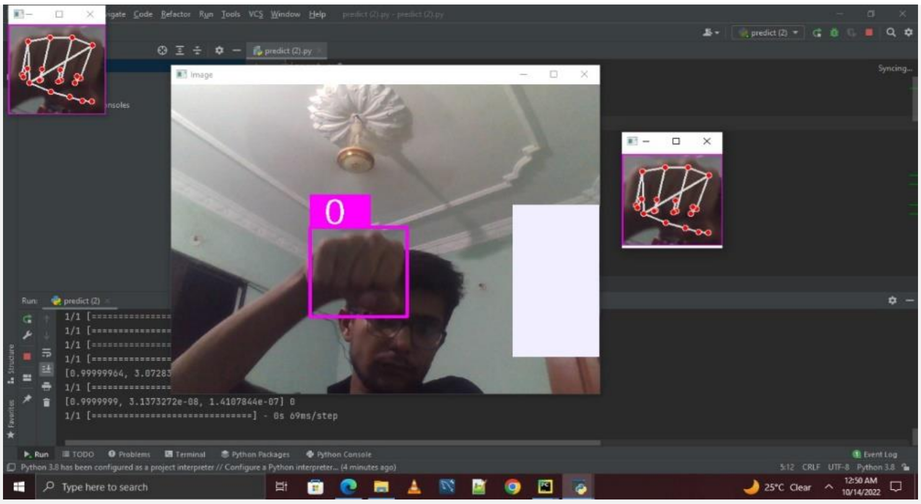

Figure. 9 Character 0 Recognition

7. Conclusion

We trained and implemented Australian Sign Language by using a Convolutional Neural Network (CNN) classifier. We were able to produce a robust model of 26 alphabets and 9 numerical digits and we got an accuracy of 92% and 95% on validation accuracy and 94% accuracy on test set. This model will help to bridge the gap between the deaf and normal people. From the results above we got that the CNN classifier gives the best result in identifying the sign language alphabets and numerical digits. This work can be further extended by making a real-time application that generates the sentences so that communication can be made easily.

REFERENCES

[1] Masood, Sarfaraz, Harish Chandra Thuwal, and Adhyan Srivastava. "American sign language character recognition using convolution neural network." In Smart Computing and Informatics: Proceedings of the First International Conference on SCI 2016, Volume 2, pp. 403-412. Springer Singapore, 2018.

[2] Mekala, Priyanka, Ying Gao, Jeffrey Fan, and Asad Davari. "Real-time sign language recognition based on neural network architecture." In 2011 IEEE 43rd Southeastern symposium on system theory, pp. 195-199. IEEE, 2011.

[3] Garcia, Brandon, and Sigberto Alarcon Viesca. "Real-time American sign language recognition with convolutional neural networks." Convolutional Neural Networks for Visual Recognition 2, no. 225-232 (2016)

[4] Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional architecture for fast feature embedding." In Proceedings of the 22nd ACM international conference on Multimedia, pp. 675-678. 2014.

[5] Weissmann, John, and Ralf Salomon. "Gesture recognition for virtual reality applications using data gloves and neural networks." In IJCNN'99. International Joint Conference on Neural Networks. Proceedings (Cat. No. 99CH36339), vol. 3, pp. 2043-2046. IEEE, 1999.

[6] Trigueiros, Paulo, Fernando Ribeiro, and Luis Paulo Reis. "Vision-based Portuguese sign language recognition system." In New Perspectives in Information Systems and Technologies, Volume 1, pp. 605-617. Springer International Publishing, 2014.

[7] Tavari, Neha V., and A. V. Deorankar. "Indian sign language recognition based on histograms of oriented gradient." International Journal of Computer Science and Information Technologies 5, no. 3 (2014): 3657-3660.

[8] Hasan, Haitham, and S. Abdul-Kareem. "RETRACTED ARTICLE: Static hand gesture recognition using neural networks." Artificial Intelligence Review 41 (2014): 147-181.

[9] Nagarajan, Sathish, and T. S. Subashini. "Static hand gesture recognition for sign language alphabets using edge oriented histogram and multi class SVM." International Journal of Computer Applications 82, no. 4 (2013).

[10] Gupta, Bhumika, Pushkar Shukla, and Ankush Mittal. "K-nearest correlated neighbor classification for Indian sign language gesture recognition using feature fusion." In 2016 International Conference on Computer Communication and Informatics (ICCCI), pp. 1-5. IEEE, 2016.

[11] Tolentino, Lean Karlo S., 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." International Journal of Machine Learning and Computing 9, no. 6 (2019): 821-827.

[12] Zhou, Ruolin, Fugang Liu, and Christopher W. Gravelle. "Deep learning for modulation recognition: A survey with a demonstration." IEEE Access 8 (2020): 67366-67376.

[13] Zhang, Lei, Yixiang Zhang, and Xiaolong Zheng. "Wisign: Ubiquitous american sign language recognition using commercial wi-fi devices." ACM Transactions on Intelligent Systems and Technology (TIST) 11, no. 3 (2020): 1-24.

[14] Mocialov, Boris, Graham Turner, Katrin Lohan, and Helen Hastie. "Towards continuous sign language recognition with deep learning.". ACM Transactions on Accessible Computing (TACCESS), 9(3) (2017), 1-27.

[15] Nimisha, K. P., and Agnes Jacob. "A brief review of the recent trends in sign language recognition." In 2020 International Conference on Communication and Signal Processing (ICCSP), pp. 186-190. IEEE, 2020.