Sign Language Alphabet Recognition Using Convolution Neural Network

Mayand Kumar
Software Engineering
Delhi Technological University
Delhi, India
mayandkumar10@gmail.com

Piyush Gupta
Software Engineering
Delhi Technological University
Delhi, India
piyush.gupta1811@gmail.com

***Abstract*— Sign language plays a vital role in the life of people who have impaired hearing and speaking inabilities. American Sign Language (ASL) alphabet recognition with the help of computer vision is a very challenging task because of the complexity in ASL signs, high inter-class similarities, constant occlusions, and large intraclass variations. This paper gives a description of a method for ASL alphabet recognition using Convolutional Neural Networks (CNN). Since a deep network is proposed for our sign language classification task, this characteristic is useful. Pre-processing operations of the MNIST data are done in the first phase. In the following phase, the various region properties of the pre-processed gesture image are computed. In the last phase, based on the properties calculated in the initial phases, we found the Accuracy of the network model with which it can recognize the sign language alphabets. Our proposed network was able to achieve an accuracy of 99.63%.**

Keywords*—American sign language (ASL), Convolution neural network (CNN), Data Augmentation, ASL number, Sign language recognition

I. INTRODUCTION

The sign language (SL) is created by hand and facial idioms to precise their outlook and thoughts of speech and hearing disabled persons with the standard (speech and hearing) people. Most normal persons may not clearly understand sign language. Therefore, there is a vast communication gap between deaf communities with the normal public. There is an inevitability of technological development support for speech impairment individuals as human translators are highly impossible to support speech impaired persons in their daily activities all the time. We can think of designing an approach that can clarify gesture signs into humanoid or machine decipherable text by the progress in science and technology. This eases the conversation between normal and impaired people. There are more than 120 distinctive sign languages used by speech impaired communities of various nations worldwide, such as American Sign Language (ASL), Indian Sign Language, Australian Sign Language, Italian Sign Language, Sri Lankan Sign Language, and many others. Over and above 60 million people globally and about 10 million people in India are using sign language as their main medium of communication.

ASL is the most widely used SL in the world and the fourth most usable linguistic in North America. Not only in the United States, but ASL is also used in Canada, Mexico, West Africa, and Asia. More than 20 other nations, including Jamaica, Panama, Thai, Malaysia, in which English is the major communication language, use ASL for their hearing-impaired community communication. Nearly two million hard of hearing people of the USA and Canada are using ASL as their primary basis of communication [1]. ASL is a broad and complicated language that usages signs made by finger and hand actions using postures of the body and expressions of the face. As ASL is seen as a precise and genuine language, it has plentiful variations, like other languages do, such as French and Spanish. ASL is an outstanding form of interaction and favorable to an enormous portion of the speech impairment population. Its foundation, existing conditions, prospect hopes, and global impact are quite amazing and eye-opening [2].

ASL includes a set of 26 gesture signs known as an American Manual Alphabet that can be used to spell out many of the English words available. The 19 various hand shapes of ASL are cast-off to make 26 American Manual Alphabets. An identical hand shape with diverse orientations is used for 'K' and 'P' letters signs. ASL also offers a set of 10 numeric gestures to sign the numbers' 0' to '9'. ASL does not comprise built-in ASL equivalents signs for accurate nouns and technical terms [3]. Along with ASL Alphabets and Numbers, there are thousands of hand and facial gesture signs available to sign the various English words. The set of 26 gesture signs of English Alphabets (A-Z) and 10 Numbers (0-9) are shown in Figure 1[1].

Motivation: American Sign Language (ASL) is a natural language that has the same linguistic characteristics as spoken languages, with grammar that is different from the English language. ASL is shown by movements of the hands and face. It is the main language of many North Americans who are deaf and have difficulty hearing and is used by some hearing people as well. Sign language recognition will make communication easier if one person isn't well-versed with the language.

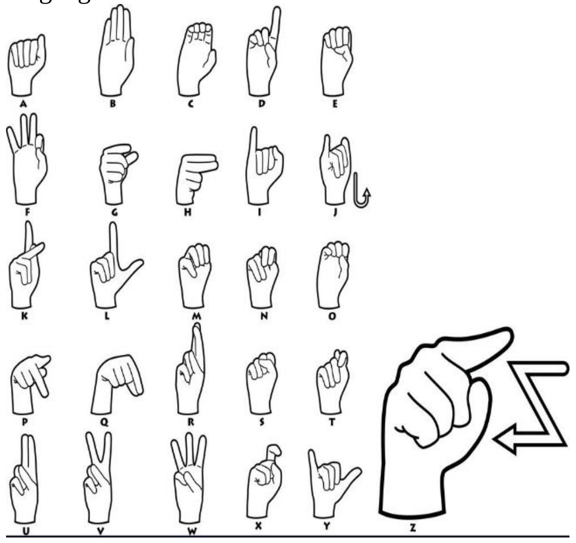


Figure 1. Hand poses are used as a basis in constructing the dataset for each letter.

II. RELATED WORK

Many approaches have been proposed in trying to solve the problem of sign language hand gesture recognition.
Few of the prior work involves [4] the use of SVM to classify South African Sign language (SASL) and [5], which used parallel hidden Markov models.
A classification approach for sign language recognition is proposed in [1]. This system recognizes 24 ASL alphabets gestures and yields an 86.67% success rate.

Pei Xu et al. [6] designed a real-time Human-Computer interaction system based on hand gestures, where each hand image was pre-processed by hand color filtering, Gaussian blurring, morphological transformations, etc. After that, a CNN was trained used to identify the gestures, and a Kalman estimator was used to move the mouse pointer in response to the gestures identified. This system achieved an average accuracy of 99.8% on 16 gestures.

Singha et al. made a database that contains 240 images for 24 signs of Indian Sign Language (ISL). Then they extracted the eigenvalues from the images in the dataset and classified them based on the Euclidean distance. This model achieved an accuracy of 97%. [7]

III. EXPERIMENTAL DESIGN

Dataset Description: Data collection is a part of research work in all the research arenas comprising sciences, scientific discipline, technology, humanity, and business furthermore. In the collection of the data procedure, methods may differ by discipline, the prominence on guaranteeing precise and authentic collection leftovers the identical. Irrespective of the study, clear-cut collection of data is a crucial stage to stabilize the integrity of the analysis. The overtly existing data collections are prohibited both in mass and class. A ceremonial progression of data collection is essential as it confirms that the data congregated are both definite and truthful. The following conclusions are contingent on arguments signified in the outcomes are valid [8].

The dataset format is designed to match closely with the classic MNIST. Each training and test case marks a label from 0 to 25 as a matched map for every alphabetic letter A-Z (and no cases for 25=Z due to gesture motions). The training data (27,455 samples) and test data (7172 samples) are approximately half the size of the standard MNIST but otherwise similar with a header row of the label, pixel1, pixel2…. pixel784 which represents a single 28x28 pixel image with grayscale values ranging 0-255. The original hand gesture image data represented multiple users repeating the gesture against entirely different backgrounds. The Sign Language MNIST data came from greatly extending the tiny variety (1704) of the colour images enclosed as not cropped around the desired hand region.

The dataset seems balanced as for each training label, and enough training examples exist shown in Figure 2 that we obtained during the visualization of the dataset.

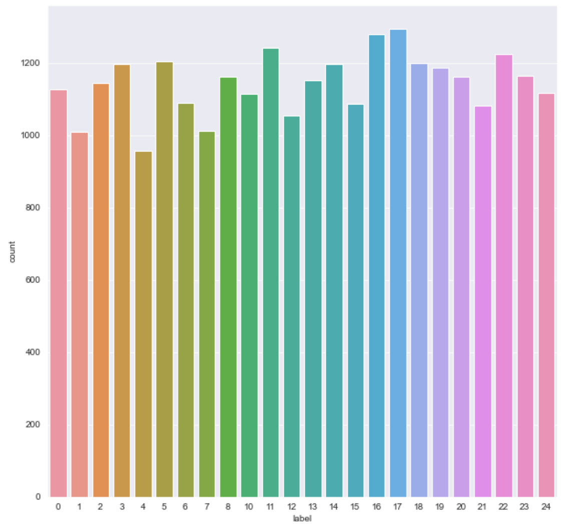


Figure 2. Number of training examples for each label

Loading and Pre-processing of Dataset: We started by importing the dataset to work upon and making it suitable to perform various algorithms to predict sign language alphabet labels correctly. In the given dataset, we already have separate training and test. We visualized our dataset, and it came out to be balanced. We also store the labels of each image, which is mapping with their respective labels in a list, and generates a random image to perform class label verification Figure 3. We performed a grayscale normalization to reduce the effect of illumination's differences. Moreover, the CNN converges faster on [0..1] data than on [0..255] Figure 4. After applying normalization, we split the training set into parts, which is the training set

[1]https://www.vectorstock.com/royalty-free-vector/sign-languagealphabet-vector-57292

and validation set. So now, we have three sets of data that is training set, validation set, and test set. In this particular problem, each of the 27455 images of the input has 784 columns. To feed this data into the CNN model, we converted the single-dimensional data into CNN accepted format of 3 dimensions. Thus, we accomplished this by converting each row of 784 columns into (28,28,1) matrix—figure 4.

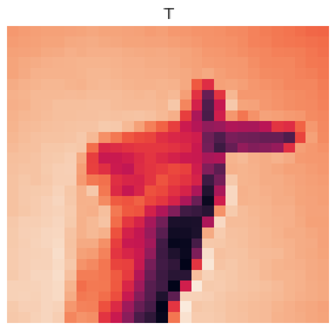


Figure 3. A Random image for class label verification.

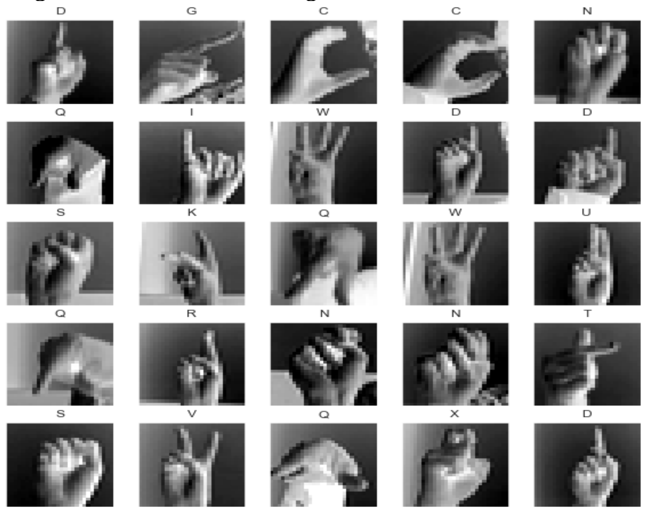


Figure 4. Grayscale images

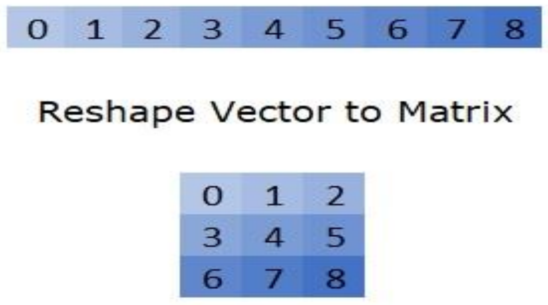


Figure 5. Reshaping of vector to matrix.

Data Augmentation: In Data augmentation from one image, we generate more images. Therefore, we can have a variety in our dataset, and it is also used to increase the size of the dataset to avoid overfitting. Our data augmentation includes random rotation ranging 10 degrees and random height and width shift in the range of 0.1, random zooming, and random flips. This, we implemented using Keras image pre-processing module.

Model Building: We build the model using the combination of convolution, max pooling, and flattening layers in Keras. We also used batch normalization and dropout layers to prevent the model from overfitting the data. We used a callback to determine if our validation set Accuracy did not increase even after 2 (patience level) consecutive epochs during the model fitting stage. Then, we will alter our learning rate by a factor of 0.5.

CNN Model: Our model is sequential in nature since we are initializing this deep learning model as a sequence of layers. The information is propagated from the input layer to the hidden layer to the output layer through the model. This network employs a mathematical operation called convolution. The primary purpose of convolution is to find features in our image using a feature detector and put them into a feature map and having them in a feature map, and it preserves the spatial relationship between pixels, which is very important for us.

The first step towards building a model for our work is to initialize the model. Since our model is sequential in nature, we initialized it as sequential. The next step is to create our first layer of CNN to perform convolution operation. We created the input layer of our CNN. We have taken 32 feature detectors of 3x3. The layer takes input into a batch of images. In our case, the batch size is 32. The shape of our image for the input layer is (28,28,1), where the height and width of the image are 28, and there is one channel as we grayscaled our images. Each feature detector produces a feature map after convolution operation over the image with an output volume of 26x26x1. We have taken the default value of padding that is 0 and stride as 1. After the convolution operation, the ReLU activation function (Figure 6.) is applied to increase non-linearity in our CNN model. If the value is greater than 0, it passes the input; otherwise, it returns 0.

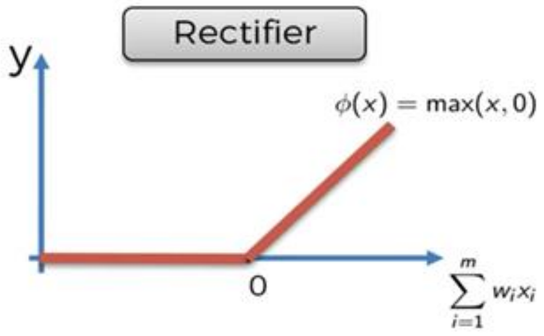


Figure 6. ReLU Activation Function

Now we have performed BatchNormalization to keep the output of the layer of the same range to get an unbiased result.After this, the output of our first convolution layer of volume 26x26x32 is passed for the max-pooling operation to the next layer. Max pooling help to get rid of unnecessary features and, moreover, account for their spatial or textual or any kind of distortion. Pooling also helps in preventing any kind of overfitting as it avoids few parameters while performing pooling operation to get a pooled matrix. Max pooling is used to focus only on the relevant information from the image by extracting only higher values from a portion of input by using an empty feature detector. Pooling is also called downsampling. We are taking stride as two and filter feature detector size as 2x2. As a result, the height and width are reduced by a factor of 2. Thus, the output volume will be 13x13x32. Now we have a second layer, which consists of 1 convolution 2d layer and one max pooling. The input of this layer is the output of the previous layer, which is 13x13x32. The kernel size is 3x3, with 64 different feature detectors. So, the output volume is 11x11x64. Now we again applied BatchNormalization and passed the output volume 11x11x64 as input for Max Pooling, and we get the output as 5x5x64. Now we have the last layer, which is the same as all previous layers consisting of 1 conv2d layer and one max pooling layer. So, the operations performed will be similar, but in the last layer, the number of feature detectors we took is 128. So, the output of the final layer will be 1x1x128. Now we will perform a flattening operation to make it suitable for ANN's input. So, after flattening, it will be converted into a 128x1 vector, and it will be fed to the dense layer (Hidden layer in ANN) with some initialized weights and bias. The output of the dense layer is passed through the ReLU activation function to increase the non-linearity in the model. Now we have used a dropout layer with a dropping rate of 25%, which means 25% of random neurons are switched off so that we can prevent overfitting of our model. Now the output of this hidden or dense layer is fed to the final or output dense layer with 25 neurons. This layer has 25 neurons representing 25 classes. The activation function here we have used is the Softmax activation function. Here we could have used the sigmoid function as the sigmoid function is the heart of the probabilistic approach, but the sigmoid function is good only when we classify between 2 classes. So, when there are more classes, softmax is used, and softmax ensures that the sum of the probability of the output is equal to one, which makes it easy to understand.

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

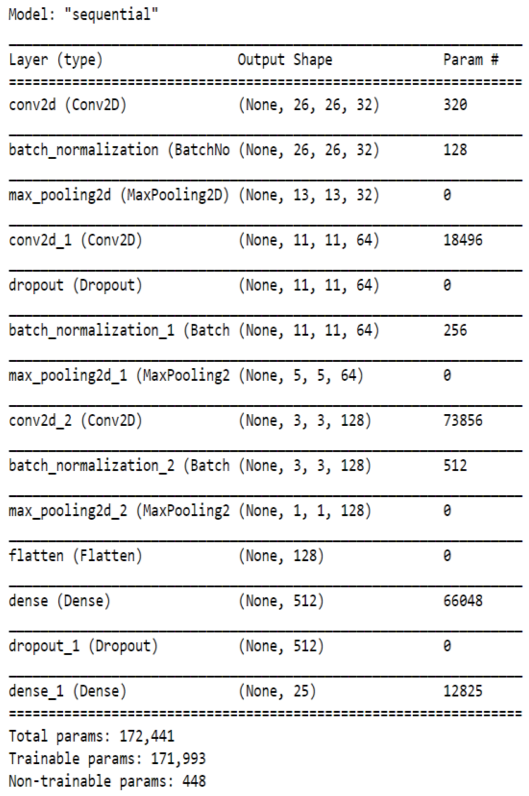
Figure 7. Equation to calculate Softmax Function.

Softmax converts the linear input data into a probability array of all 25 classes, and the probabilities are checked with the actual output, and the loss is calculated using sparse categorical cross-entropy. Here we have used sparse categorical cross-entropy because we have labeled as targets.

Finally, we compile our model, and to minimize our cost function, we use 'adam' optimizer (stochastic Gradient Descent Method) to attain a global minimum. 'adam' optimizer is used with a learning rate of 0.001 and values of b1=0.9 and b2=0.999.

During the backpropagation in the model, weights in the dense layer, as well as weights present in the feature detector changes using the local gradient if not useful for the model.

The total trainable parameters to be calculated are 171,993, as seen in Figure 8., representing each layer with its output shape and trainable parameters.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 26, 26, 32)        320
_________________________________________________________________
batch_normalization (BatchNo (None, 26, 26, 32)        128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 13, 13, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 11, 11, 64)        18496
_________________________________________________________________
dropout (Dropout)            (None, 11, 11, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 11, 11, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 5, 5, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 3, 3, 128)         73856
_________________________________________________________________
batch_normalization_2 (Batch (None, 3, 3, 128)         512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 1, 1, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 128)               0
_________________________________________________________________
dense (Dense)                (None, 512)               66048
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 25)                12825
=================================================================
Total params: 172,441
Trainable params: 171,993
Non-trainable params: 448
```

Figure 8.

Independent and Dependent Variables:
Independent variable- number of images which is 27455, pixel distribution of images which is stored in numpy array.
Dependent variables are- feature map after each convolution layer output, volume of feature map which is input to the next layer, probabilities of labels which is used for classification.

Parameters and Hyper Parameters:
Parameters like kernel size, stride of filter, padding, number of features/filters, filter size, activation function, connection between two layers, pooling type, number of labels.
Hyper parameters like coefficients in learning rate, dropout rate, batch size, number of hidden and dense layer in architecture, number of epochs.

Performance Measure:
Performance measures are used to check the correctness of our model. The majors we are going to use are Accuracy on both training and validation dataset, loss in both validation and training dataset, Precision, Recall and F1 measure of all 25 classes.

Validation Method: We are using Holdout validation technique to evaluate our model. We initially have a training and testing set. Our dataset contains 34,627 images. 20% of total images (7172) are present in our testing set and rest in the training set. We split our training set into two parts, one for training our model and another part which is a validation set for validating our model. Later we will analyse our model based on predictions predicted by our trained model on the test set and obtain its accuracy, precision, recall, F1 Score and AUC Score to see its performance.

IV. RESULTS AND ANALYSIS

The network was tested on 7,712 images that it did not see during training of the network. The accuracy, precision, recall, AUC Score, and F1 Score of the classification was determined. Table. 1 shows the result.

We trained our model on 20 epochs with a batch size of 512.The maximum accuracy obtained on the training set was 99.60% and in validating set was 99.95%. Epoch Vs Accuracy graph is shown in Figure 9 and Figure 10 for training and validation accuracy, training and validation loss respectively.

Table 1.
Accuracy, Precision, Recall, AUC Score, and F1 Score obtained from the test.

	Accuracy	Precision	Recall	AUC Score	F1 Score
Ours	0.9963	0.99	1.00	0.9981	1.00

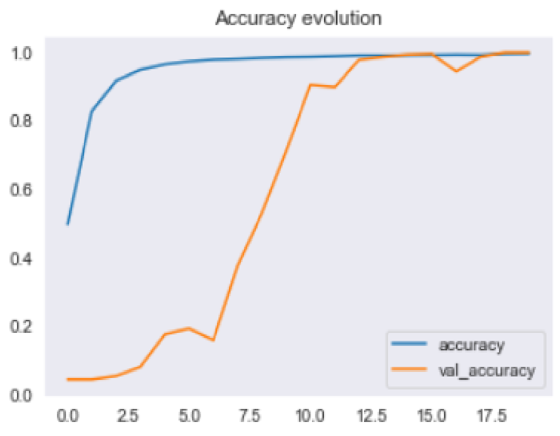


Figure 9

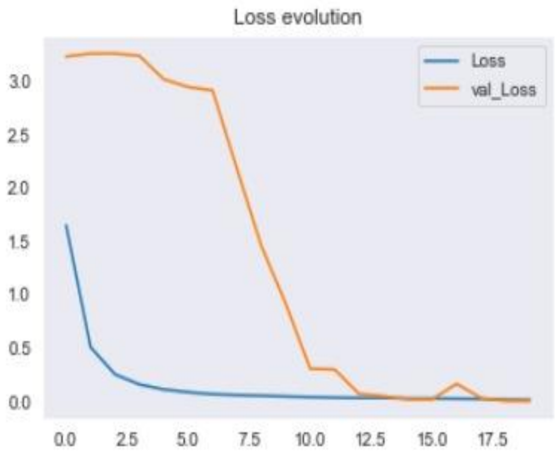
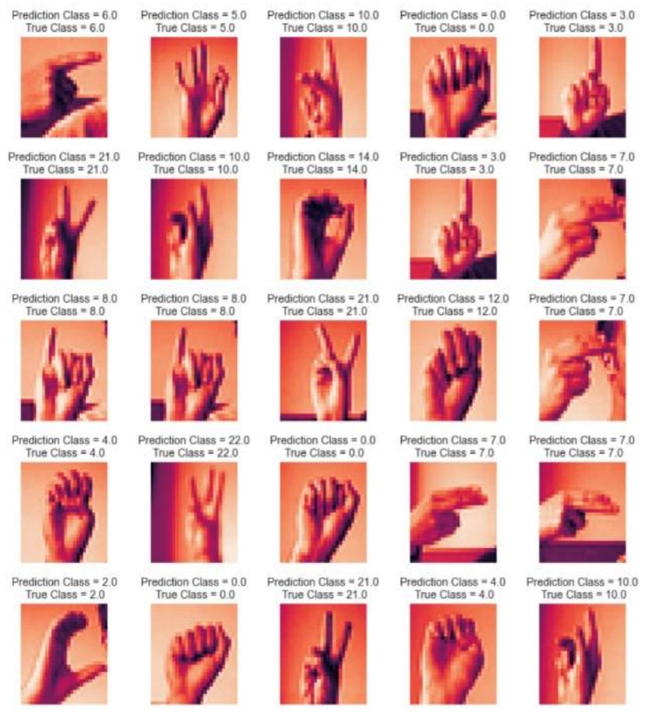


Figure 10

Aside from accuracy, Precision, recall, F1 Score, and AUC Score, a confusion matrix was also obtained to properly contrast the predicted letter and the ground truth. Each entry represents the frequency of the predicted letter in comparison to its ground truth. This is illustrated in Figure 11. The y-axis represents the true level while x-axis represents the predicted level.
From the confusion matrix, it was determined that the majority of the predictions are correct. Some letters had difficulty in prediction, particularly due to the similarity in the appearances of the letters. For example, O and S are confused with each other since the fingers are positioned in the same way with only a slight difference in the way the fingers are clenched, and the position of the thumb. Since we are using MNIST dataset, most of the predictions are correct. Some of the predictions of our network is shown in the Figure 12.



Prediction Class = 6.0 True Class = 6.0
Prediction Class = 5.0 True Class = 5.0
Prediction Class = 10.0 True Class = 10.0
Prediction Class = 0.0 True Class = 0.0
Prediction Class = 3.0 True Class = 3.0
Prediction Class = 21.0 True Class = 21.0
Prediction Class = 10.0 True Class = 10.0
Prediction Class = 14.0 True Class = 14.0
Prediction Class = 3.0 True Class = 3.0
Prediction Class = 7.0 True Class = 7.0
Prediction Class = 8.0 True Class = 8.0
Prediction Class = 8.0 True Class = 8.0
Prediction Class = 21.0 True Class = 21.0
Prediction Class = 12.0 True Class = 12.0
Prediction Class = 7.0 True Class = 7.0
Prediction Class = 4.0 True Class = 4.0
Prediction Class = 22.0 True Class = 22.0
Prediction Class = 0.0 True Class = 0.0
Prediction Class = 7.0 True Class = 7.0
Prediction Class = 7.0 True Class = 7.0
Prediction Class = 2.0 True Class = 2.0
Prediction Class = 0.0 True Class = 0.0
Prediction Class = 21.0 True Class = 21.0
Prediction Class = 4.0 True Class = 4.0
Prediction Class = 10.0 True Class = 10.0

Figure 12. Predictions of the network on the test set.

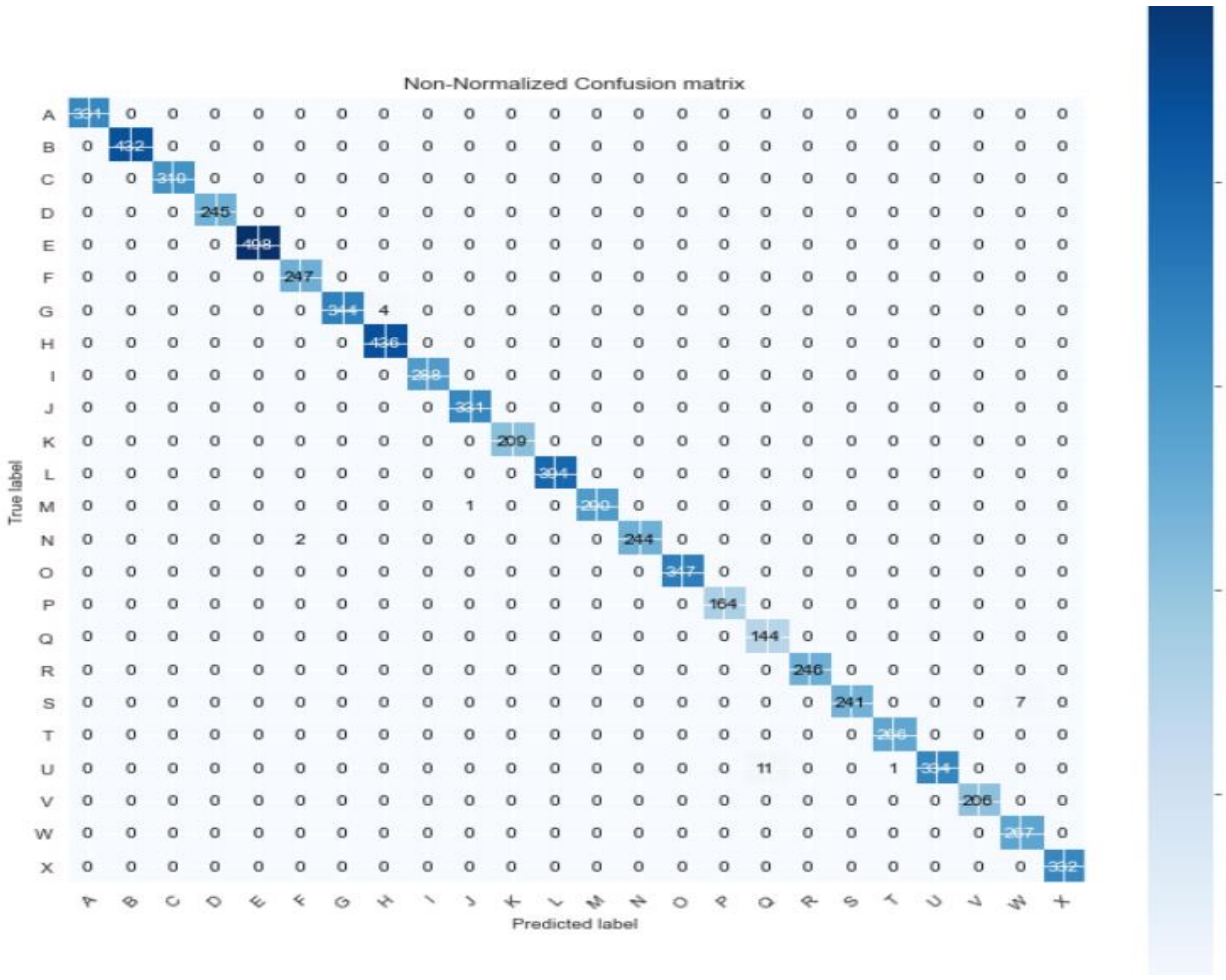
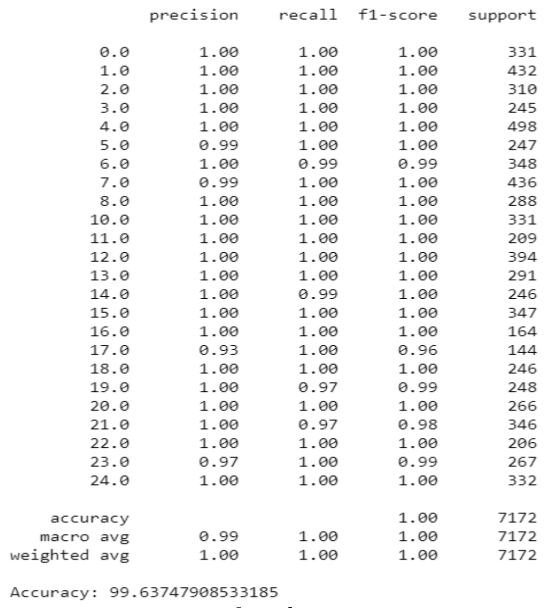


Figure 11 Confusion matrix showing the frequency of the predicted letters in contrast to the ground truth, where highlighted numbers indicate the higher frequencies. The x-axis represents the predicted letters while the y-axis represents the ground truth.

The Classification report obtained in the end is shown in Figure 13. And precision, Recall, F1 Score for each label is also shown.



```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       331
         1.0       1.00      1.00      1.00       432
         2.0       1.00      1.00      1.00       310
         3.0       1.00      1.00      1.00       245
         4.0       1.00      1.00      1.00       498
         5.0       0.99      1.00      1.00       247
         6.0       1.00      0.99      0.99       348
         7.0       0.99      1.00      1.00       436
         8.0       1.00      1.00      1.00       288
        10.0       1.00      1.00      1.00       331
        11.0       1.00      1.00      1.00       209
        12.0       1.00      1.00      1.00       394
        13.0       1.00      1.00      1.00       291
        14.0       1.00      0.99      1.00       246
        15.0       1.00      1.00      1.00       347
        16.0       1.00      1.00      1.00       164
        17.0       0.93      1.00      0.96       144
        18.0       1.00      1.00      1.00       246
        19.0       1.00      0.97      0.99       248
        20.0       1.00      1.00      1.00       266
        21.0       1.00      0.97      0.98       346
        22.0       1.00      1.00      1.00       206
        23.0       0.97      1.00      0.99       267
        24.0       1.00      1.00      1.00       332

    accuracy                           1.00      7172
   macro avg       0.99      1.00      1.00      7172
weighted avg       1.00      1.00      1.00      7172

Accuracy: 99.63747908533185
```

Figure 13. Classification Report

V. CONCLUSION AND FUTURE WORK

This paper exhibits an approach based on convolutional neural networks for recognizing sign language alphabet. Our proposed CNN Model was able to perform with 99.63% accuracy on our test dataset. Initially when we were using only one convolutional layer there was overfitting in the dataset and to overcome this issue of overfitting, we added two more layers so that we can drop some parameters and still preserve the essential features of images. Moreover, we also used dropout layer, BatchNormalization and data augmentation to overcome the same and achieve a better performance of our network.

Future study may extend our work to accept video frames to include letters j and z in the classification so that more varied inputs can be processed and understood by the network. Furthermore, there is a need for a large public dataset for automatic sign language recognition to utilize new deep learning techniques and a better way to benchmark performance. There are currently very few common sets of data being used, and most of the time researches being conducted use a subset of this. This makes it more difficult to compare different works - a dataset might have a different effect on a network compared to another.

REFERENCES

[1] Ganesh Krishna Sharma, Srinath S, "Classification approach for Sign Language Recognition", ICS, Image Processing, Communication & Automation, 2017.

[2] Srinath S, Shivashankara S, "A comparative Study of Various Techniques and Outcomes of Recognizing American Sign Language: A Review", (IJSRET), Vol. 6, Issue 9, pp.1013-1023, 2017.

[3] Srinath S, Shivashankara S, "A Review on Vision Based American Sign Language Recognition, its Techniques, and Outcomes", 7th IEEE (CSNT-2017), pp.293-299, 2017.

[4] S. Naidoo, C. Omlin, and M. Glaser, "Vision- based static hand gesture recognition using support vector machines," University of Western Cape, Bellville, 1998.

[5] C. Vogler and D. Metaxas, "Parallel hidden markov models for american sign language recognition," in Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on, vol. 1. IEEE, 1999, pp. 116–122.

[6] Pei Xu, A real time hand gesture recognition and human-computer interaction system, In: Proceeding of the Computer Vision and Pattern Recognition, 2017

[7] J. Singha and K. Das, "Indian Sign Language Recognition Using eigen-value Weighted Euclidean Distance Based Classification Technique", International Journal of Advanced Computer Science and Applications, vol. 4, no. 2, 2013.

[8] Dr. Roger Sapsford, Victor Jupp, "Data Collection and Analysis", 2nd Edition, Sage Publishing Ltd, 2006.

[9] Wenjin Taoa, Ming C. Leua, Zhaozheng Yinb, "American Sign Language Alphabet Recognition Using Convolutional Neural Networks with Multiview Augmentation and Inference Fusion " Network", LATEX, 2018

[10] Neel Kamal Bhagat, Vishnusai Y, Rathna G N, "Indian Sign Language Gesture Recognition using Image Processing and Deep Learning" Network", IEEE, 2019