CSE/ECE 343 : Machine Learning Project Proposal
Sign language to Text conversion

Bhavesh Sood - 2019355
Ajeet Yadav - 2019010
Vishwajeet Kumar - 2019128
Group No: 31

Abstract

For interaction between normal people and Deaf & Dumb people a language barrier is created as sign language structure which is different from normal text. So they depend on vision based signs and gestures for communication and interaction. If there is a common interface that converts the sign language to text, then gestures can be easily understood by other people. Research has been made for a vision based interface system where deaf and mute people can enjoy communication without really knowing each other's language. Github

1. Introduction

Our aim is to develop a user-friendly human computer interface (HCI) where the computer understands the human sign language. There are various sign languages all over the world, we will be using American Sign Language (ASL) for this project.

2. Literature Survey

1. This project used American sign language to convert into speech and text using the techniques of image segmentation and feature detection. The system goes through various phases such as data capturing, sensor, image segmentation, feature detection and extraction etc.[1]

2. This project is based on Creating a desktop application that captures a person signing gestures for American sign language (ASL), and translate it into corresponding text and speech in real time.[2]

3. This project builds a machine learning model which can classify the various hand gestures used in sign language. In this model, classification machine learning algorithms are trained using a set of image data.[3]

3. Dataset

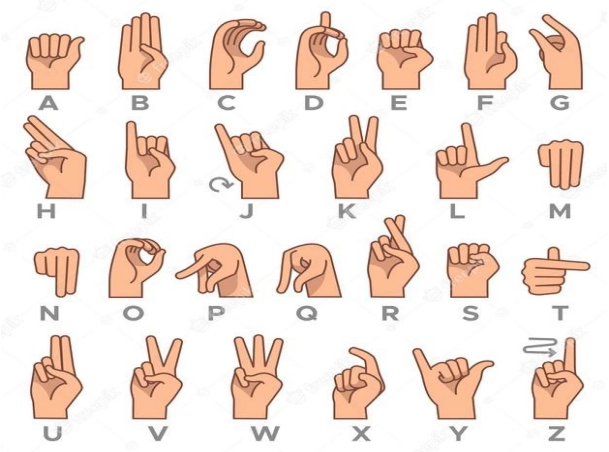


Figure 1. All signs and their corresponding letters

Our dataset is a large database of drawn representation of different gestures for American Sign Letters.[4] The database contains 27,455 training images and 7172 testing images each of size 28x28. In our data we have 784 columns of pixel1, pixel2,..., pixel784 which represents a single 28X28 image. These images contains matrices of pixel values and each pixel value is in range 0-255. Target column has labels integers between 1-26 corresponding to english Alphabets A to Z. Our dataset does not contains any data for 9 = J and 25 = Z because in sign languae these alphabets needs motion.

All these pixels values can be presented directly to our model but this can result in challenges during modeling such as slower than expected training of the model. Instead, we believe it can be great benefit in preparing the pixel values before doing any modeling such as standardization.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance). Standardization scales each input

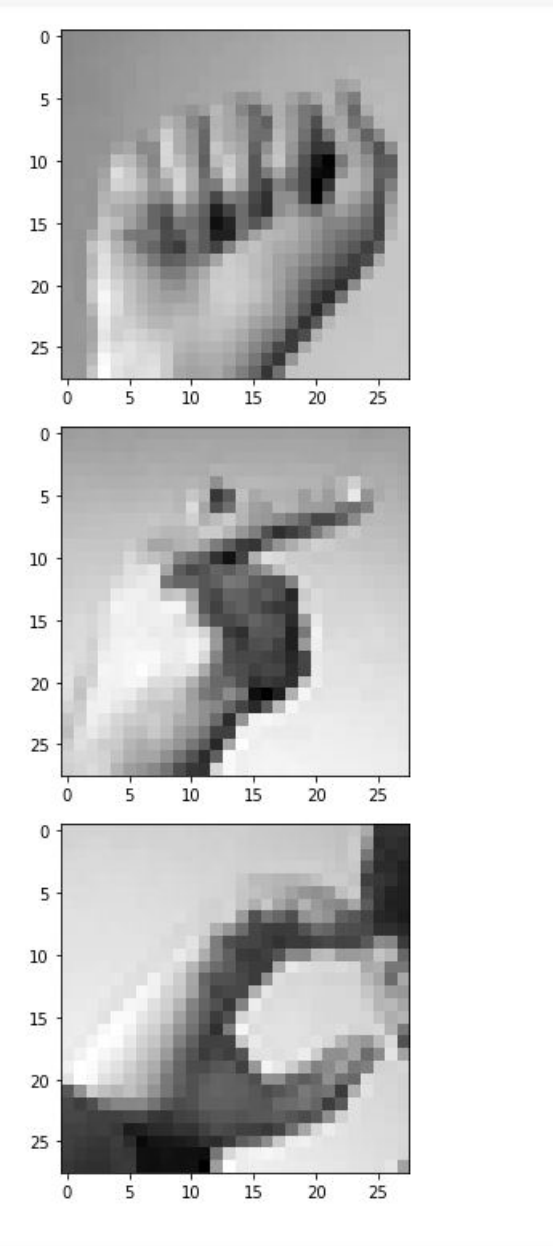


Figure 2. Example pictures of the dataset

variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

Each pixel value are the features of the dataset and since each pixel is important for an image so we need not to do any feature selection/reduction. Our dataset does not have any missing, NaN, noisy or inconsistent value so we feel we need not to do any kind of Data-cleaning.

4. Methodology

The goal is to make a model such that the image of the sign gets correctly converted to one of the 26 letters of the alphabet.The problem is to classify the images(input) into one-of the 26 labels(26 Alphabets of English language).

4.1. Logistic Regression

we started with one of the basic classification techniques. We tried training a simple Logistic Regression model. Logistic regression trains the data using sigmoid hypothesis function and gradient descent algorithm. We trained the Logistic regression model of sklearn on train our data. We set the penalty field of parameter to none so that there is no regularisation. Our problem is multiclass classification. For multiclass classification this model uses one-vs-rest scheme. We trained the model for 400 iterations.

4.1.1 Logistic Regression with L2 regularisation

After training logistic regression normally we tried to add some penalty to decrease the variance. So we first trained our data with L2 regularisation(Ridge regression). We used the same linearRegression model of sklearn but changed its penalty parameter(which is l2 by default). We trained this data also for 400 iterations.

4.1.2 Logistic Regression with L1 regularisation

After trying out L2 regression we tried L1 regularisation(Lasso regression) as it is robust to outliers. This time we changed the penalty field of logisticRegression model to 'l1' and trained the training data again for 400 iterations.

4.2. Decision Tree

We then tried to make a Decision Tree for classifying the images. Decision tree is a very direct, rule based way to classify the data. It makes a tree considering some column as its root and starts to classify the data by selecting some other column as its child. We used the 'DecisionTreeClassifier' from sklearn to train our data. It uses GINI Index(as default) for Impurity measurement and choses the column for root on its basis. As we were trying pure decision tree so we left the max_depth parameter of tree as it is(default value NONE). Which means nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

After training the model on train data we calculated the accuracy of our model on the test data. The test data is also taken from kaggle.

4.3. Ensembling on DT model (Random Forest Classifier)

Our Decision Tree was not giving good accuracy, so we decided to try the ensembling technique. Since RF classifier is an ensemble method that trains several decision trees in parallel with bootstrapping followed by aggregation, so we trained a RF model with default parameters. Our test accuracy got significantly improved as compared to Decision Tree from 43.83% to 80% .

4.3.1 Hyper-parameter tuning of RF model

So far we have received the best accuracy of 80% with the RF model, we further tried to improve the accuracy by tuning the hyper-parameters of it. Following were the hyper-parameters and their values from which we tried to find the best params using grid search CV:

'n_estimators': [80, 100, 120, 200, 300],
'criterion': ['entropy','gini'],
'max_depth': [2, 4, 10, 18, 30]

After running for 5 cross-validations, grid Search CV gave the following combination of best hyper-parameters, n_estimators=300, max_depth=30 and criterion as "entropy". With these parameters, the accuracy got improved from 80% to 81.49% .

5. SVM

Before moving to Neural Networks we tried SVM, which is an l2 norm soft margin classifier. SVM is a supervised ML algorithm that can be used for both classification and regression. We used the 'rbf' kernel for this , with no limit on max iterations since SVM always converges. The main advantage of SVM is that it is highly effective in highly dimensional spaces. We trained the model on trained data, tested on test data and got accuracy of 84.88%. Results of SVM model on our classification problem was best so far.

6. MLP

After trying above common ML algorithms, we decided to try Artificial Neural Networks for our classification Problem. Artificial Neural Network is a connections of neurons, replicating the structure of human brain. Each connection of neuron transfers information to another neuron. Inputs are fed into first layer of neurons which processes it and transfers to another layer of neurons called as hidden layers. After processing of information through multiple layers of hidden layers, information is passed to final output layer.

After trying many combinations of hidden layer sizes, solvers, max_iters and activation functions and other hyper parameter tuning we created a MLP classifier with hidden layers sizes as (600, 650, 700) with activation function 'relu' and after training and testing, the model gave an accuracy of 84.54% on test data. We calculated the confusion matrix and found weighted precision, recall and F1 score values for further confirmation and those all values were about 0.85. So we reached the conclusion that 85% is the nearby best that we can get from MLP.

7. CNN

Unlike regular Neural Networks, in the layers of CNN, the neurons are arranged in 3 dimensions: width, height, depth. The neurons in a layer will only be connected to a small region of the layer (window size) before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would have dimensions (number of classes), because by the end of the CNN architecture we will reduce the full image into a single vector of class scores. In our model we have used three -2D convolutional layers each followed by maxpool to reduce the spatial dimensions. Then we compiled the model with tuned parameters and trained it keeping a validation set in hand of 8:2 to keep on a check over over fitting problem. Here is the summary of our CNN model.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 26, 26, 32)        320

 max_pooling2d_6 (MaxPooling  (None, 13, 13, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 11, 11, 128)       36992

 max_pooling2d_7 (MaxPooling  (None, 5, 5, 128)        0
 2D)

 conv2d_8 (Conv2D)           (None, 3, 3, 256)         295168

 max_pooling2d_8 (MaxPooling  (None, 1, 1, 256)        0
 2D)

 flatten_2 (Flatten)         (None, 256)               0

 dense_8 (Dense)             (None, 64)                16448

 dense_9 (Dense)             (None, 128)               8320

 dense_10 (Dense)            (None, 128)               16512

 dense_11 (Dense)            (None, 25)                3225

=================================================================
Total params: 376,985
Trainable params: 376,985
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. Summary of CNN

8. Results

8.1. Accuracies

1. Logistic Regression

   (a) Simple: 65.29%

   (b) Ridge: 70.65%

   (c) Lasso: 63.30%

2. Decision Tree : 43.83%

3. Random Forest : 81.49%

4. Support Vector Machine : 84.88%

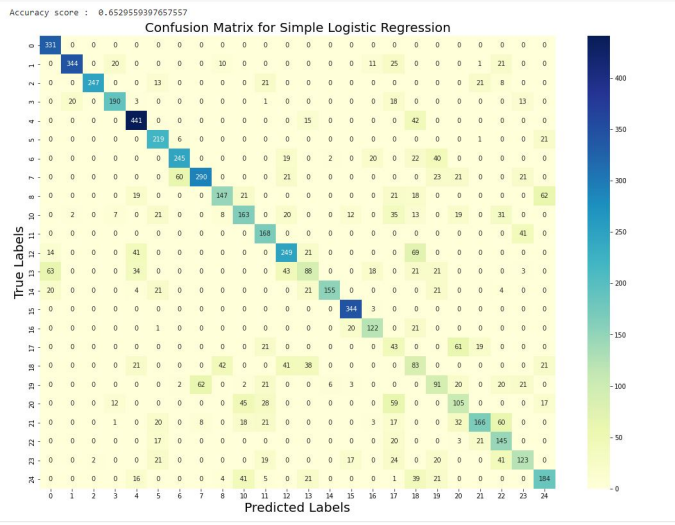
5. MLP classifier : 84.53%

6. CNN classifier : 95.50%

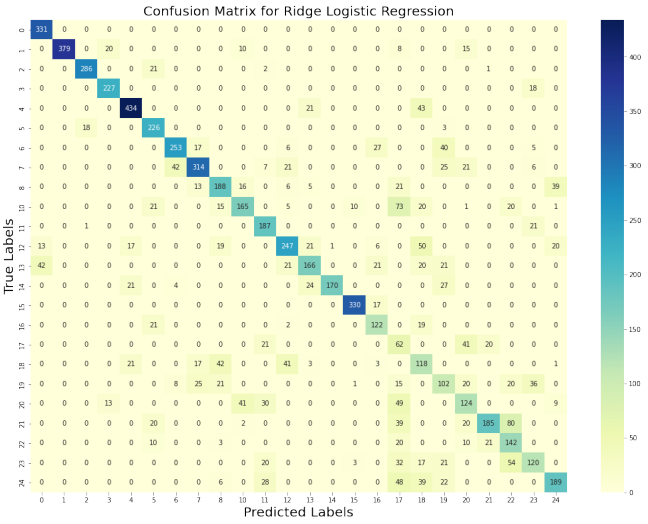
8.2. Confusion Matrices

1. Logistic Regression

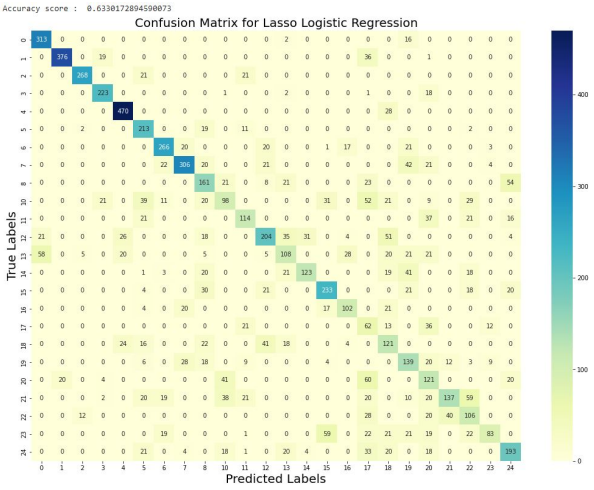
   (a) Simple



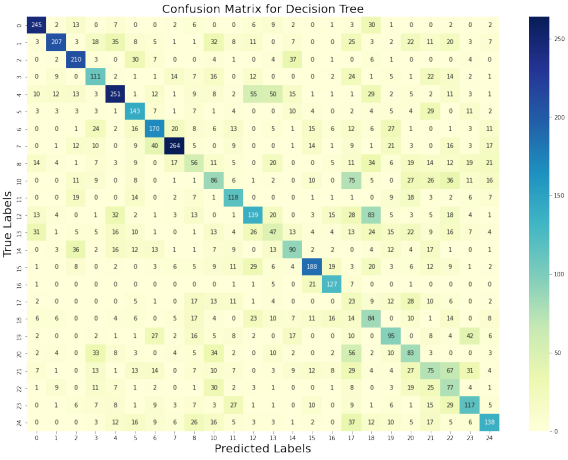
   (b) Ridge



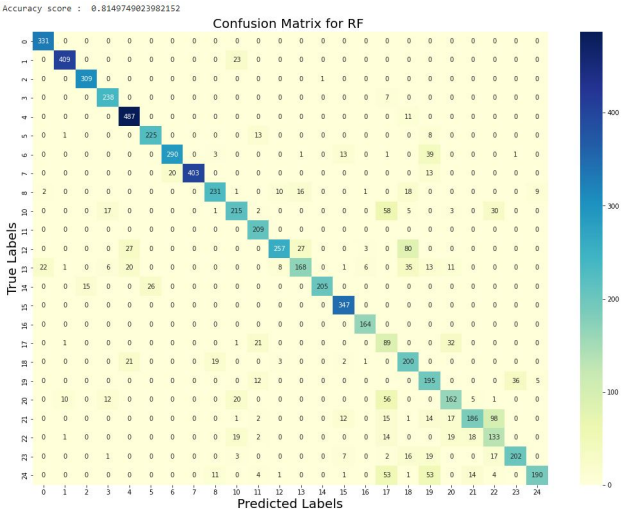
   (c) Lasso



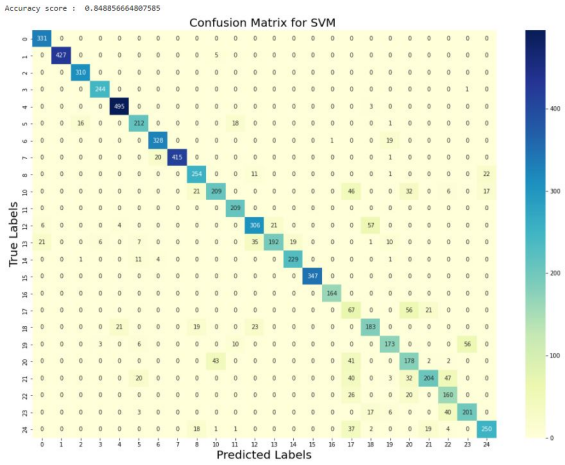
2. Decision Tree



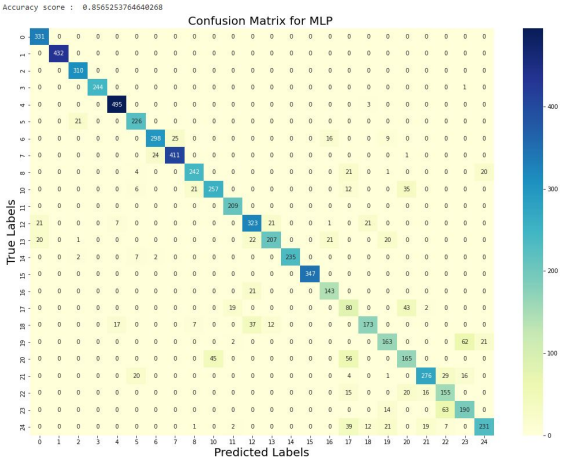
3. Random Forest



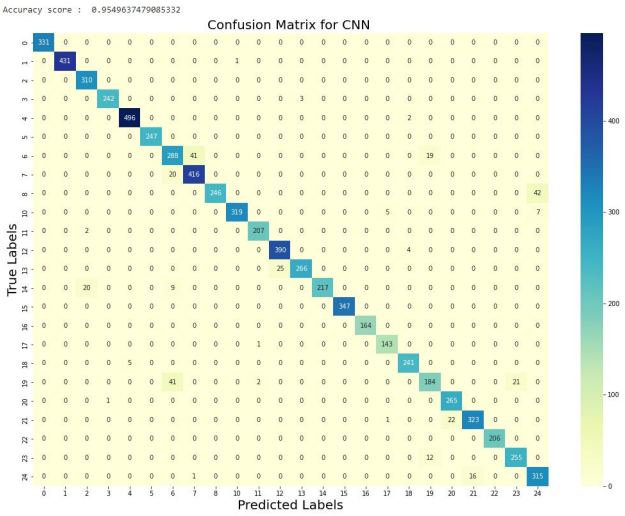
4. Support Vector Machine



5. MLP Classifier



6. CNN Classifier



8.3. Precision | Recall | F1 Score

1. Logistic Regression
   (a) Simple
      i. Precision : 0.67
      ii. Recall : 0.65
      iii. F1 :0.65
   (b) Ridge
      i. Precision : 0.74
      ii. Recall : 0.71
      iii. F1 :0.71
   (c) Lasso
      i. Precision : 0.66
      ii. Recall : 0.63
      iii. F1 :0.64
2. Decision Tree
   (a) Precision : 0.46
   (b) Recall : 0.44
   (c) F1 : 0.44
3. Random Forest
   (a) Precision : 0.84
   (b) Recall : 0.81
   (c) F1 : 0.82
4. Support Vector Machine
   (a) Precision : 0.86

(b) Recall : 0.85

(c) F1 : 0.85

5. MLP Classifier

(a) Precision : 0.86

(b) Recall : 0.85

(c) F1 : 0.85

6. CNN Classifier

(a) Precision : 0.96

(b) Recall : 0.95

(c) F1 : 0.95

9. Conclusion

Analysing all the models with their accuracy and loss curve for training, validation and testing data, CNN was giving a test set accuracy of 95.5% which is good enough. Other models were also giving decent accuracy but were not above 85% even after hyper parameter tuning.

Also in CNN we tried several combinations of different parameters and with an learning rate of the order 0.001 it was converging really fast. While in case of 0.0001 the accuracy was dropping to 92%. So we took the middle value of learning rate = 0.0005 and trained the data for 20 epochs. This provided us with a good accuracy and loss curve.

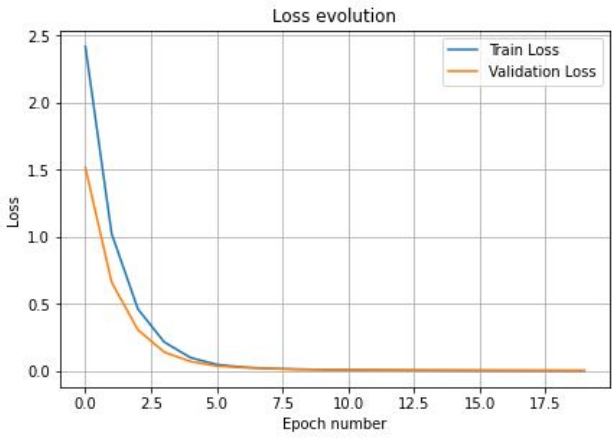


Figure 4. Loss vs Epochs curve

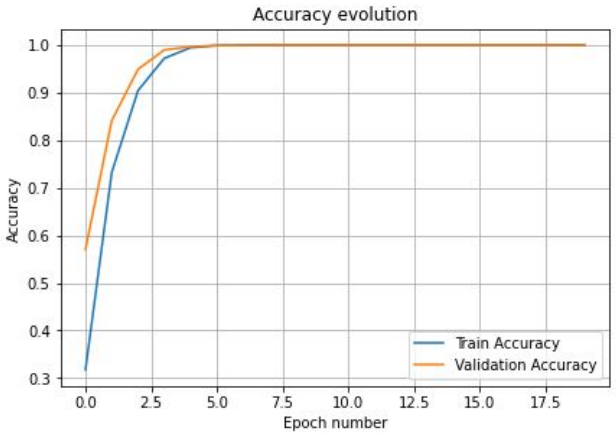


Figure 5. Accuracy vs Epochs curve

Figure 6. Original picture with full dimensions

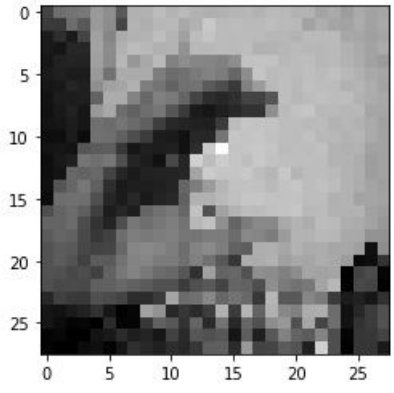


Figure 7. Gray Scaled picture of 28x28 dimension

For example we took above image(fig 6) as input from live camera and converted it to grayscale image(fig 7) which is shown below original image. Our code predicted the label 2 for above image which is the label for C as labelling starts from 0() for A, 1 for B and 2 for C).

So finally after trying all the models with different combinations of parameters we reached the conclusion that CNN is the best suitable model for our problem with an accuracy of 95.5%.

10. Work Distribution

1. Bhavesh: Imported the dataset, filtered the data into labels and features, trained the initial model for Logistic Regression(all types) and Decision Trees , significantly contributed in making the report in LaTeX. Trained the SVM model , and worked on the final CNN model tuning. Also worked on the real time picture translation using OpenCV and converted the image to data to predict it's class from the model.

2. Vishwajeet: Preprocessing of data. Accuracy and scores calculation, trained models for MLP classifier and CNN, And helped in reading input through OpenCv. Helped in presentation and Report.

3. Ajeet: Visualized the pixel-array to actual image, classification report calculation for the models, ensembling on DT models (RF model), RF model accuracy improvement by Hyper-parameter tuning using GridSearchCV, report writing and presentation-slides making.

References

[1] Victoria A. Adewale and Dr. Adejoke O.Olamiti . Conversion of Sign Language To Text And Speech Using Machine Learning Techniques. [1]. 1

[2] Ankit Ojha, Ayush Pandey, Shubham Maurya, Abhishek Thakur, and Dr. Dayananda P. Sign language to text and speech translation in real time using convolutional neural network. 2014. [2]. 1

[3] Muskan Dhiman. Sign language recognition. 2017. [3]. 1

[4] Sign language mnist. [4]. 1