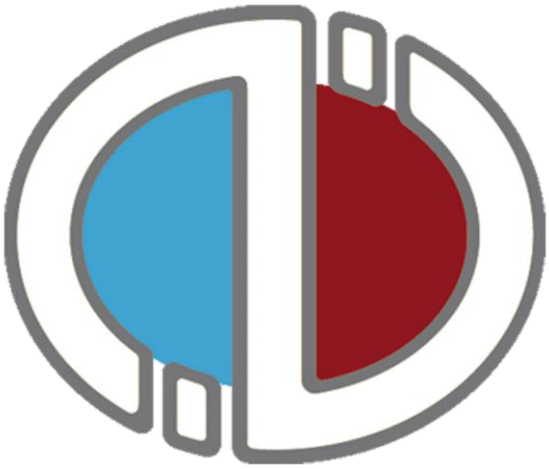
Sign Language Recognition

PROJECT REPORT

Jama Hussein MOHAMUD | Machine Vision | 08/05/2018



Department of Electrical & Electronic Engineering

Anadolu University

Eskisehir, Turkey

Year: 2018

Abstract

Inability to speak is considered to be true disability. People with this disability use different modes to communicate with others, there are number of methods available for their communication one such common method of communication is sign language. Developing sign language application for deaf people can be very important, as they'll be able to communicate easily with even those who don't understand sign language. Our project aims at taking the basic step in bridging the communication gap between normal people, deaf and dumb people using sign language.

The main focus of this work is to create a vision based system to identify sign language gestures from the video sequences. The reason for choosing a system based on vision relates to the fact that it provides a simpler and more intuitive way of communication between a human and a computer. In this report, 24 different gestures have been considered.

INTRODUCTION

Motion of any body part like face, hand is a form of gesture. Here for gesture recognition we are using image processing and computer vision. Gesture recognition enables computer to understand human actions and also acts as an interpreter between computer and human. This could provide potential to human to interact naturally with the computers without any physical contact of the mechanical devices. Gestures are performed by deaf and dumb community to perform sign language. This community used sign language for their communication when broadcasting audio is impossible, or typing and writing is difficult, but there is the vision possibility. At that time sign language is the only way for exchanging information between people. Normally sign language is used by everyone when they do not want to speak, but this is the only way of communication for deaf and dumb community. Sign language is also serving the same meaning as spoken language does. This is used by deaf and dumb community all over the world but in their regional form like ISL, ASL. Sign language can be performed by using Hand gesture either by one hand or two hands. It is of two type Isolated sign language and continuous sign language. Isolated sign language consists of single gesture having single word while continuous ISL or Continuous Sign language is a sequence of gestures that generate a meaningful sentence.

In this project two different approach is applied, one is deep neural networks and the other one is bag of words model. We will first explain all the steps for each approach, then see the results.

Block Diagram

ASL (American Sign Language) Dataset is used in this project. I have 4972 sample images for training, 24 images for testing and 24 different classes which represents 26 gestures, where J and Z represents two more gestures.

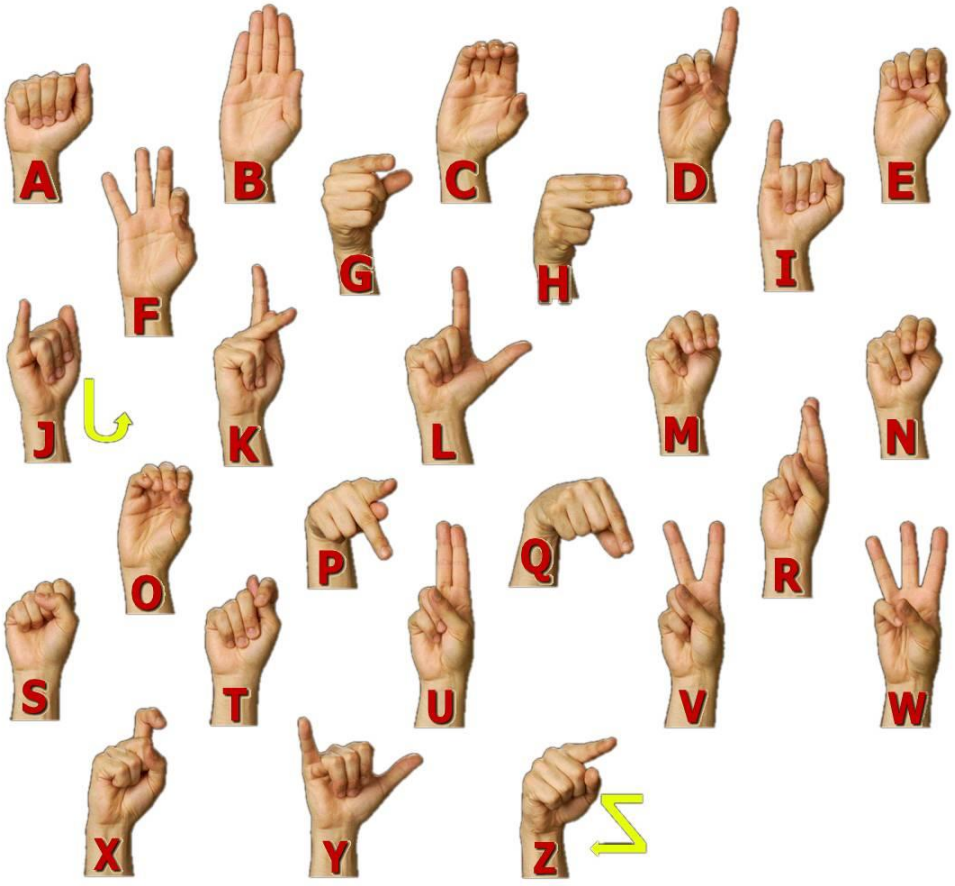


Fig 1: American Sign Language

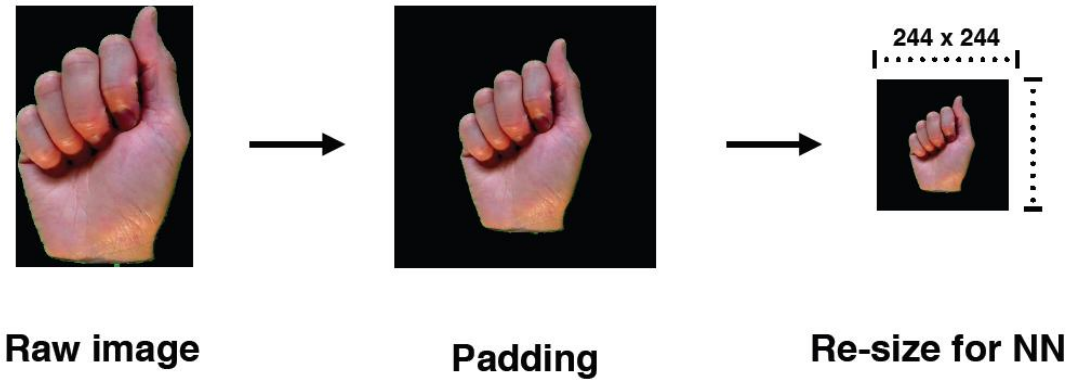
Preprocessing

In the preprocessing step we are applying three steps: cropping, padding and resizing.

It is very import to preprocess our images before we feed to our deep neural network, because our classifier will perform better compared when it is not preprocessed.

Changing background color: background of images are changed to black in order to be aware of artifacts. (Optional)

Cropping (Uniform aspect ratio): One of the first steps is to ensure that the images have the same size and aspect ratio. Most of the neural network models assume a square shape input image, which means that each image need to be checked if it is a square or not, and cropped appropriately. Cropping can be done to select a square part of the image, as shown. While cropping, we usually care about the part in the center. (Optional)



Padding: Padding is to add extra pixels outside the image. And zero padding means every pixel value that you add is zero. If zero padding = 1, there will be one pixel thick around the original image with pixel value = 0. Every time we use the filter (a.k.a. kernel) to scan the image, the size of the image will go smaller and smaller. We don't want that, because we want to preserve the original size of the image to extract some low level features. Therefore, we will add some extra pixels outside the image!

Image resizing: Once we've ensured that all images are square (or have some predetermined aspect ratio), it's time to resize each image appropriately. We've decided to have images with width and height of 244 pixels. There are a wide variety of up-scaling and down-scaling techniques and we usually use a library function to do this for us.

Feature Extraction

A feature basically represents a very small region in an image. This region, also called "patch", should be unique and discriminative. Feature representation methods deal with the problem of representing these patches with a vector of numbers. These vectors are called feature descriptors. One of the most famous descriptors is Scale-Invariant Feature Transform (SIFT). SIFT converts each patch into a 128-dimensional vector. After this step, each image is a collection of vectors of the same dimension, where the order of different vectors is not important.

In this project SURF feature extraction is used. Which is abbreviated as Speeded-Up Robust features. As the name suggest it is a speeded-up version of SIFT.

K means clustering is also used as a feature extraction, although it is not feature extraction method but it helps us to cluster descriptors from SURF in to K clusters. More formally, we first, extract descriptors from the image on a grid or around detected keypoints. Next, for each descriptor extracted its nearest neighbor in the dictionary is computed. Finally, a histogram of length k where the i'th value is the frequency of the i'th dictionary word is constructed.

Algorithms

As I have mentioned in section 2, two different models are used for this classification problem. Each model is trained with different classification algorithm, but let us discuss the two that provides us the best accuracy. However both of these algorithms needs to re-trained in order to achieve better performance.

Convolutional neural networks: Convolutional Neural Networks are a powerful artificial neural network technique. These networks preserve the spatial structure of the problem and were developed for object recognition tasks such as handwritten digit recognition. They are popular because people are achieving state-of-the-art results on difficult computer vision and natural language processing tasks.

Convolutional Neural Networks expect and preserve the spatial relationship between pixels by learning internal feature representations using small squares of input data. Feature are learned and used across the whole image, allowing for the objects in the images to be shifted or translated in the scene and still detectable by the network.
It is this reason why the network is so useful for object recognition in photographs, picking out digits, faces, objects and so on with varying orientation.
In summary, below are some benefits of using convolutional neural networks:

- They use fewer parameters (weights) to learn than a fully connected network.
- They are designed to be invariant to object position and distortion in the scene.
- They automatically learn and generalize features from the input domain.

Convolutional layers are comprised of filters and feature maps

- **Filters**

The filters are the "neurons" of the layer. They have input weights and output a value. The input size is a fixed square called a patch or a receptive field. If the convolutional layer is an input layer, then the input patch will be pixel values. If the deeper in the network architecture, then the convolutional layer will take input from a feature map from the previous layer.

- **Feature Maps**

The feature map is the output of one filter applied to the previous layer.
A given filter is drawn across the entire previous layer, moved one pixel at a time. Each position results in an activation of the neuron and the output is collected in the feature map. You can see that if the receptive field is moved one pixel from activation to activation, then the field will overlap with the previous activation by (field width – 1) input values.

- **Zero Padding**

The distance that filter is moved across the input from the previous layer each activation is referred to as the stride. If the size of the previous layer is not cleanly divisible by the size of the filters receptive field and the size of the stride then it is possible for the receptive field to attempt to read off the edge of the input feature map. In this case, techniques like zero padding can be used to invent mock inputs for the receptive field to read.

- **Pooling Layers**

The pooling layers down-sample the previous layers feature map. Pooling layers follow a sequence of one or more convolutional layers and are intended to consolidate the features learned and expressed in the previous layers feature map. As such, pooling may be consider a technique to compress or generalize feature representations and generally reduce the overfitting of the training data by the model. They too have a receptive field, often much smaller than the convolutional layer. Also, the stride or number of inputs that the receptive field is moved for each activation is often equal to the size of the receptive field to avoid any overlap. Pooling layers are often very simple, taking the average or the maximum of the input value in order to create its own feature map.

- **Fully Connected Layers**

Fully connected layers are the normal flat feed-forward neural network layer. These layers may have a non-linear activation function or a softmax activation in order to output probabilities of class predictions. Fully connected layers are used at the end of the network after feature extraction and consolidation has been performed by the convolutional and pooling layers. They are used to create final non-linear combinations of features and for making predictions by the network.

This is our fully connected CNN layers:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_89 (Conv2D)           (None, 32, 32, 32)        896
_________________________________________________________________
dropout_48 (Dropout)         (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_90 (Conv2D)           (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d_36 (MaxPooling (None, 32, 16, 16)        0
_________________________________________________________________
conv2d_91 (Conv2D)           (None, 64, 16, 16)        18496
_________________________________________________________________
dropout_49 (Dropout)         (None, 64, 16, 16)        0
_________________________________________________________________
conv2d_92 (Conv2D)           (None, 64, 16, 16)        36928
_________________________________________________________________
max_pooling2d_37 (MaxPooling (None, 64, 8, 8)          0
_________________________________________________________________
conv2d_93 (Conv2D)           (None, 128, 8, 8)         73856
_________________________________________________________________
dropout_50 (Dropout)         (None, 128, 8, 8)         0
_________________________________________________________________
conv2d_94 (Conv2D)           (None, 128, 8, 8)         147584
_________________________________________________________________
max_pooling2d_38 (MaxPooling (None, 128, 4, 4)         0
_________________________________________________________________
flatten_16 (Flatten)         (None, 2048)              0
_________________________________________________________________
dropout_51 (Dropout)         (None, 2048)              0
_________________________________________________________________
dense_34 (Dense)             (None, 1024)              2098176
_________________________________________________________________
dropout_52 (Dropout)         (None, 1024)              0
_________________________________________________________________
dense_35 (Dense)             (None, 512)               524800
_________________________________________________________________
dropout_53 (Dropout)         (None, 512)               0
_________________________________________________________________
dense_36 (Dense)             (None, 24)                12312
=================================================================
Total params: 2,922,296
Trainable params: 2,922,296
Non-trainable params: 0
```

K nearest neighbor (KNN): this another supervised machine learning algorithm that is used in BOW model. In other words we are given dataset that has both training samples and labels, and we would like to capture the relationship between them.

In the classification setting, the K-nearest neighbor algorithm essentially boils down to forming a majority vote between the K most similar instances to a given “unseen”

observation. Similarity is defined according to a distance metric between two data points. A popular choice is the Euclidean distance given by

$$d(x, x') = \sqrt{(x_1 - x'_1)^2 + (x_2 - x'_2)^2 + \ldots + (x_n - x'_n)^2}$$

More formally, given a positive integer **K**, an unseen observation **x** and a similarity metric **d**, KNN classifier performs the following two steps:

- It runs through the whole dataset computing d between x and each training observation. We'll call the K points in the training data that are closest to x the set A. Note that **K** is usually odd to prevent tie situations.
- It then estimates the conditional probability for each class, that is, the fraction of points in A with that given class label. (Note I(x) is the indicator function which evaluates to 1 when the argument x is true and 0 otherwise)

An alternate way of understanding KNN is by thinking about it as calculating a decision boundary (i.e. boundaries for more than 2 classes) which is then used to classify new points.

Next, let us summarize the steps we took in each model.

Approach 1 (Deep neural networks):

- **Pre- Processing**: Since we already explain each preprocessing step. Let us only list their names here, to indication that they are part of approach one.
    1. Changing background color (Optional)
    2. Cropping (Uniform aspect ratio) (Optional)
    3. Padding
    4. Image resizing
- **Convolutional neural network**

Approach 2 (Bag of words model):

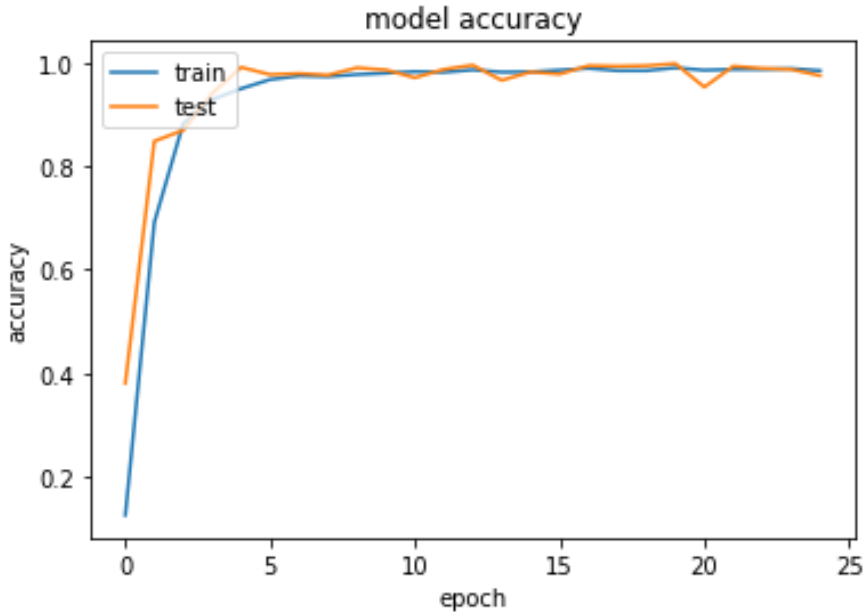
- SURF feature extraction
- Kmeans Clustering: we cluster the set of descriptors to k clusters. The cluster centers act as our dictionary's visual words.
- KNN (K nearest neighbors)

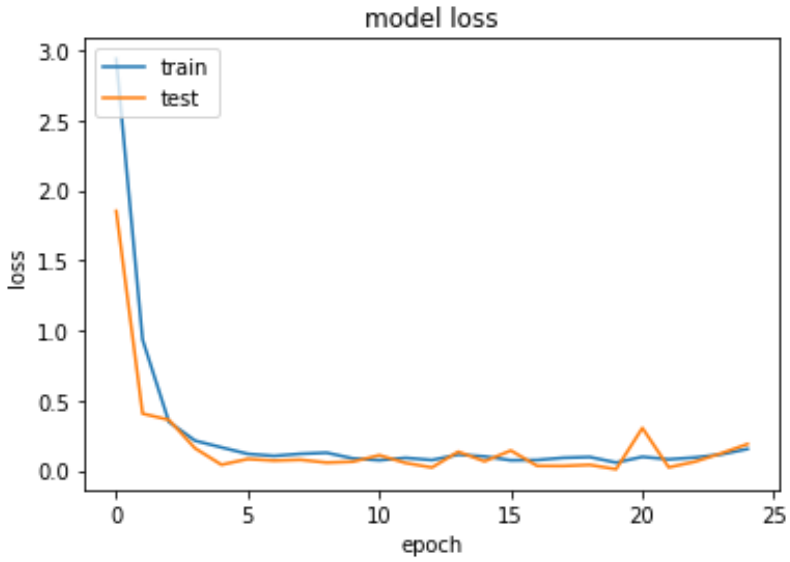
Results

In Deep neural network algorithms are extremely very good at image processing and computer vision problems. Very high accuracy is achieved using ASL dataset.
We trained our model up to 25 epochs and got

```
1. Test Loss: 0.18818815050918103
2. Test accuracy: 0.9748743718592965
```

Below are the loss and accuracy plots.

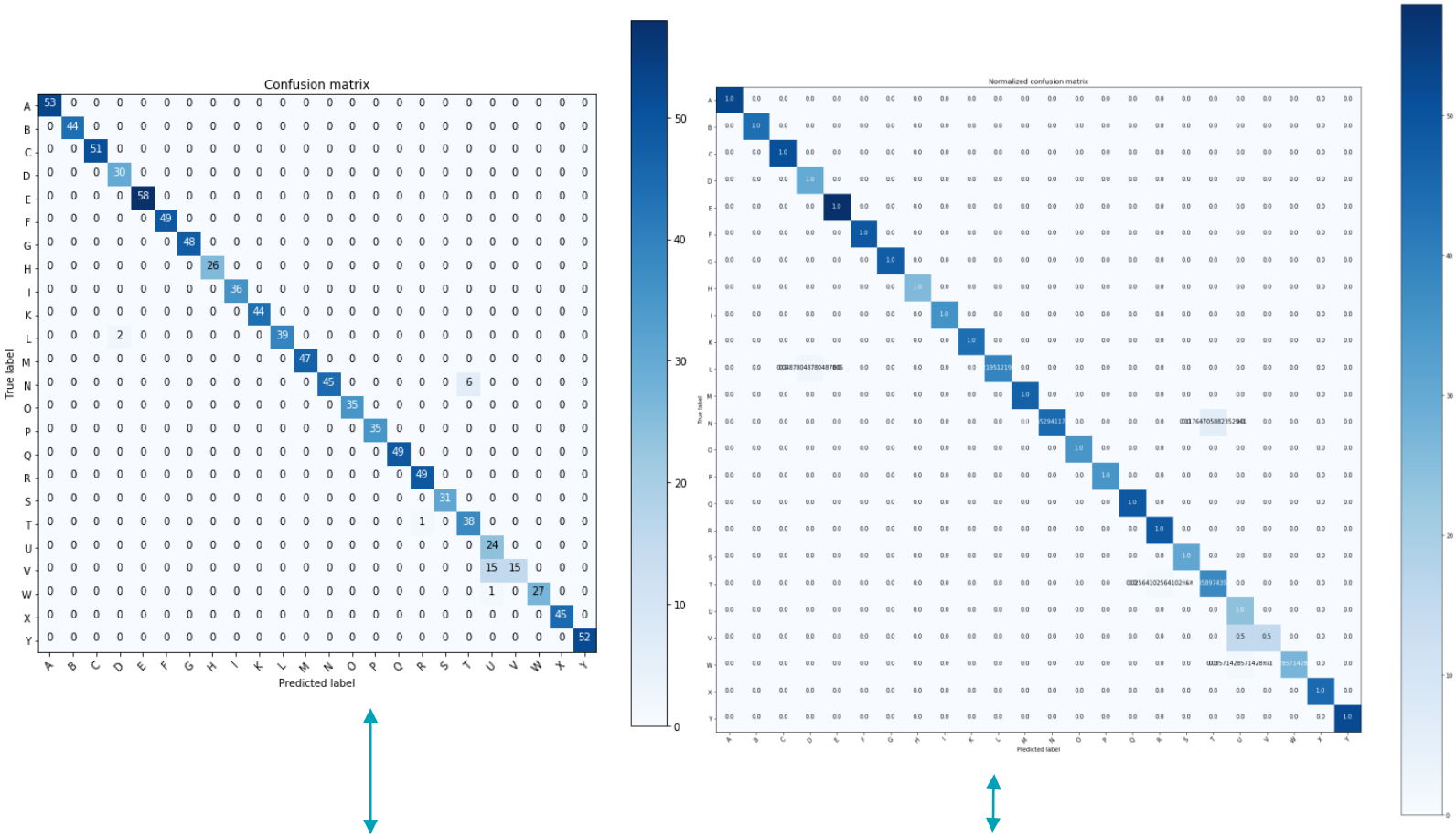




The main problem with classification accuracy is that it hides the detail you need to better understand the performance of your classification model. There are two examples where you are most likely to encounter this problem:

- When you are data has more than 2 classes. With 3 or more classes you may get a classification accuracy of 80%, but you don't know if that is because all classes are being predicted equally well or whether one or two classes are being neglected by the model.
- When your data does not have an even number of classes. You may achieve accuracy of 90% or more, but this is not a good score if 90 records for every 100 belong to one class and you can achieve this score by always predicting the most common class value.

Classification accuracy can hide the detail you need to diagnose the performance of your model. But thankfully we can tease apart this detail by using a **confusion matrix.**



Confusion matrix without normalization

Normalized confusion matrix

In order to have a good confidence level, I, Lastly, calculated precision, recall and F1 score. Let us first define these terms.

Precision is the number of True Positives divided by the number of True Positives and False Positives. Put another way, it is the number of positive predictions divided by the total number of positive class values predicted. It is also called the Positive Predictive Value (PPV).
Precision can be thought of as a measure of a classifiers exactness. A low precision can also indicate a large number of False Positives.

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate.
Recall can be thought of as a measure of a classifiers completeness. A low recall indicates many False Negatives.

The F1 Score is the 2*((precision*recall)/ (precision + recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.
And these are our result

```
    precision    recall  f1-score   support

A       1.00      1.00      1.00        53
B       1.00      1.00      1.00        44
C       1.00      1.00      1.00        51
D       0.94      1.00      0.97        30
E       1.00      1.00      1.00        58
F       1.00      1.00      1.00        49
G       1.00      1.00      1.00        48
H       1.00      1.00      1.00        26
I       1.00      1.00      1.00        36
K       1.00      1.00      1.00        44
L       1.00      0.95      0.97        41
M       1.00      1.00      1.00        47
N       1.00      0.88      0.94        51
O       1.00      1.00      1.00        35
P       1.00      1.00      1.00        35
Q       1.00      1.00      1.00        49
R       0.98      1.00      0.99        49
S       1.00      1.00      1.00        31
T       0.86      0.97      0.92        39
U       0.60      1.00      0.75        24
V       1.00      0.50      0.67        30
W       1.00      0.96      0.98        28
X       1.00      1.00      1.00        45
Y       1.00      1.00      1.00        52

Total   0.98      0.97      0.97       995
```

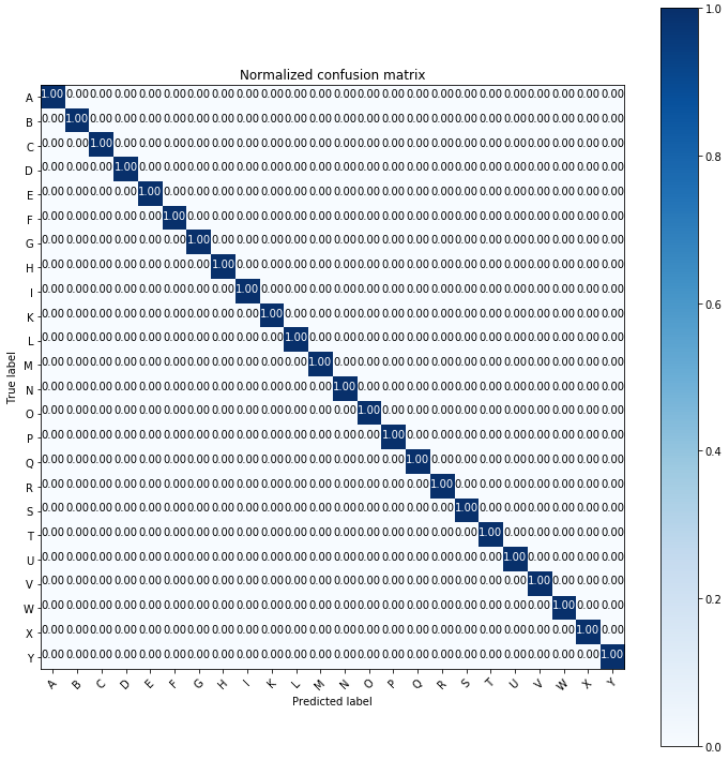
BOW (bag-of-words) results:

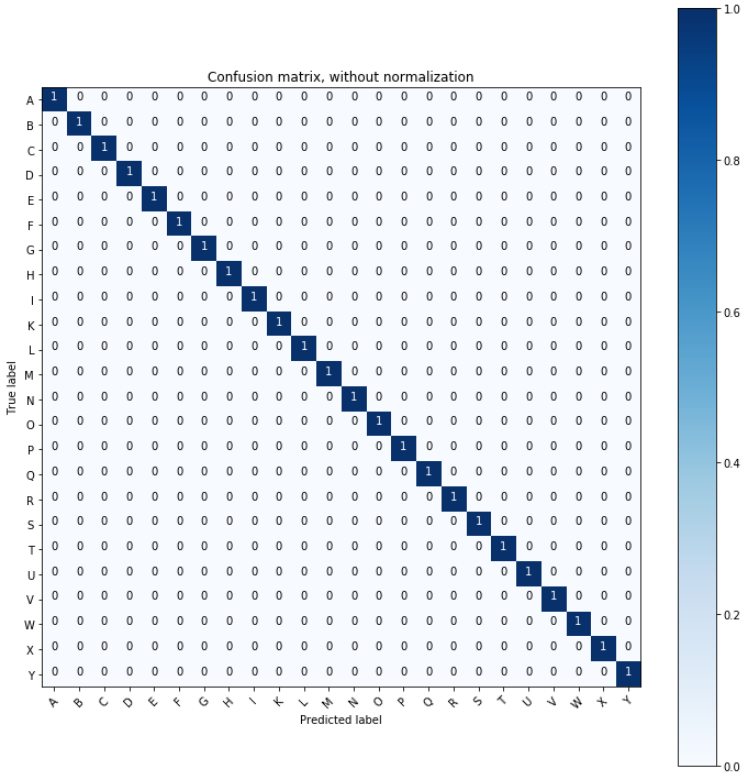
For BOW model, we computed the general accuracy and plotted the confusion matrix. Surprisingly, I have obtained too high accuracy

```
1.  Training Accuracy:  1.0
2.  Test Accuracy:  1.0
```

We could have said that it is due to Overfitting, but it doesn't seem to be. Since, confusion matrix show good results.

The reason I got high test accuracy is that I only used 24 sample images as test set. But if we could validate with more images, as we did CNN, we would have had less accuracy.





Conclusions and Discussion

In summary, we have implemented both CNN and BOW models and seen how they perform in ASL dataset. However, for BOW we need to increase the test dataset so that we see how it performs in unseen dataset. For CNN we achieved incredible results and the only thing left is to add real-time part.

Reference

[1]. http://cs229.stanford.edu/proj2011/ChenSenguptaSundaram-SignLanguageGestureRecognitionWithUnsupervisedFeatureLearning.pdf

[2]. http://cs229.stanford.edu/proj2011/KurdyumovHoNg-SignLanguageClassificationUsingWebcamImages.pdf

[3]. https://pdfs.semanticscholar.org/6321/a370d0398fbe6cde1899d6c0ad0992ca333b.pdf