Mini Project Report

American Sign Language Detection

Mihir Joshi Ashlesh Gupte Mansi Maurya

Under Guidance of

Prof Amol Kalugude

APSIT

Statement of Purpose:

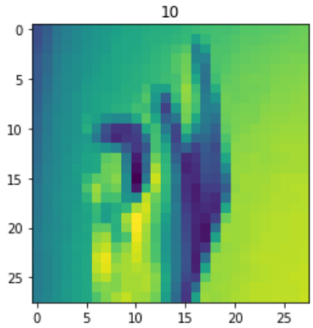
Develop a Machine Learning model for detection of American Sign Language Detection and Conversion. MNIST dataset is used for training the model using Keras and Tensor flow. By applying neural network algorithms we develop a model which can identify sign language with an accuracy of 70%. We use a classification algorithm to classify the image into 26 classes.

Results on sample data:

```
In [0]: img = test_X[6]
        test_img = img.reshape((1,784))
        img_class = model.predict_classes(test_img)
        prediction = img_class[0]
        classname = img_class[0]
        print("Class: ",classname)

        Class:  10

In [0]: img = img.reshape((28,28))
        plt.imshow(img)
        plt.title(classname)
        plt.show()
```



When a test image is passes to the model it classifies it and gives the class (label) of that image which is passed.

Dataset used:

MNIST DATASET

https://www.kaggle.com/datamunge/sign-language-mnist

Literature Referenced:

International Conference on Advances in Computing, Communication Control and Networking (ICACCCN2018)

Dynamic Tool for American Sign Language Finger Spelling Interpreter

ISBN:978-1-5386-4119-4/18

Technologies Used:

Keras

Python

Tensorflow

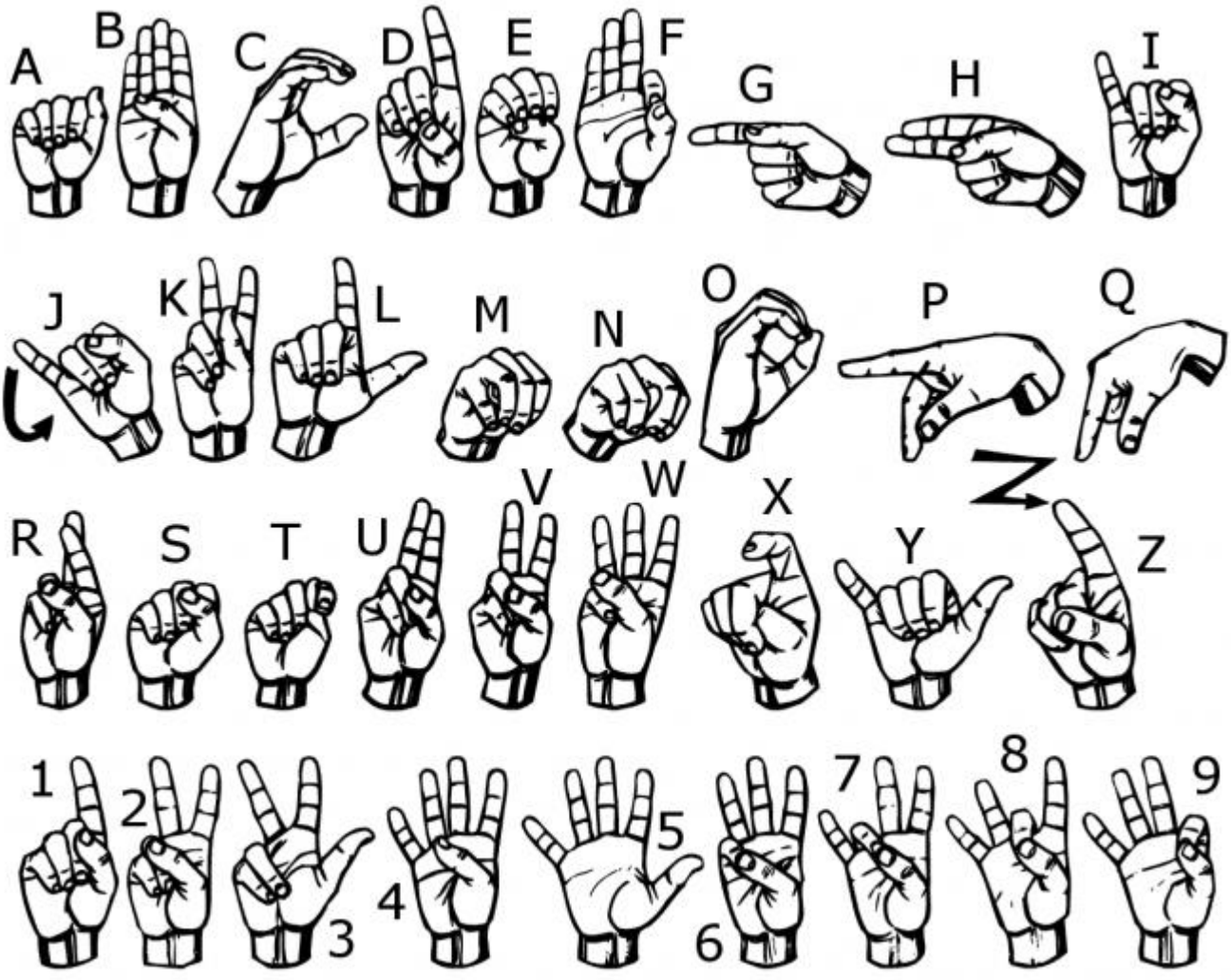
Google Colab

Machine Learning Schema

1) Dataset

We will use MNIST (Modified National Institute of Standards and Technology)dataset.

Basically, our dataset consists many images of 24 (except J and Z) American Sign Language alphabets. Each image has size 28x28 pixel which means total 784 pixels per image.



- **Loading the dataset to Colab**

Our dataset is in CSV (Comma-separated values) format. train_X and test_X contain the values of each pixel. train_Y and test_Y contains the lable of image. You can use the following code to see the dataset:

- **Pre-processing**

train_X and test_X consists an array of all the pixel pixel values. We have to create an image from these values. Our image size is 28x28 hence we have to divide the array into 28x28 pixel groups. To do that we will use the following code:

2) Build and Train the Model

We will use CNN (Convolutional Neural Network) to recognise the alphabets. We are going to use keras.

As you can observe, like any other CNN our model consists couple of Conv2D and MaxPooling layers followed by some fully connected layers (Dense).

The first Conv2D (Convolutional) layer takes input image of shape (28, 28, and 1). The last fully connected layer gives us output for 26 alphabets.

We are using a Dropout after 2nd Conv2D layer to regularise our training.

We are using softmax activation function in the final layer. Which will give us probability for each alphabet as an output.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 8)         80
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 8)         0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 14, 14, 16)        1168
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 16)        0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 16)          0
_________________________________________________________________
dense_1 (Dense)              (None, 3, 3, 128)         2176
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_2 (Dense)              (None, 26)                29978
=================================================================
Total params: 33,402
Trainable params: 33,402
Non-trainable params: 0
_________________________________________________________________
```

We are using SGD optimiser to compile our model.