Kathmandu University

Department of Computer Science and Engineering

Dhulikhel, Kavre

A Mid-term Project Report

on

"Traffic Sign Recognition System"

COMP 484

(For partial fulfillment of 4th Year/ 1st Semester in Computer Engineering)

Submitted by:

Junth Basnet (05)

Sandip Dulal (63)

Abin Sainju (65)

Submitted to:

Dr. Bal Krishna Bal

Associate Professor

Department of Computer Science and Engineering

Submission Date:

December 30, 2019

Project Title: Traffic Sign Recognition System

Proposed method and Experiments done so far

Data Summary & Exploration

Throughout this section, we used the Numpy, Pandas, and Matplotlib libraries to explore and visualize the traffic signs data set. Our 'train' folder contains 43 folders each representing a different class. The range of the folder is from 0 to 42. With the help of the OS module, we iterate over all the classes and append images and their respective labels in the data and labels list.

Data Shape and Size

Size of the training set	**29,209 (56.34%)**
Size of the validation set	**10,000 (19.29%)**
Size of the test set	**12,630 (24.36%)**
Shape of traffic sign image	**(32, 32, 3)**
Number of unique classes/labels	**43**

Data Visualization

Before designing the neural network, we felt it was important to visualize the data in various ways to gain some intuition for what the model will "see." This not only informs the model structure and parameters, but it also helps in determining the types of preprocessing operations to be applied to the data.

Image Samples and Corresponding Labels

Figure 1 shows the sample of the original images before they undergo any preprocessing. Overall, the image quality is good and the labels make intuitive sense. However, immediately we noticed a few things we'll want to adjust during preprocessing: Many of the signs are hard to recognize because the images are dark and have low contrast. There is little variation in the sign shape and viewing angle. Most of the pictures are taken with a straight-on view of the sign, which is good for the core data set. However, in real life, signs are viewed from different angles. The signs are void of any deformations or occlusions. Again, this is good because we need a clean set of training samples, but in real life, signs are sometimes damaged, vandalized, or only partially visible. Essentially, we want the model to recognize signs even when the shape is distorted, much like humans can. So, augmenting the training set with a variety of distortions is important.

Class/Label Distribution

The distribution of training image samples per class is not uniform. The largest classes have 10x the number of traffic sign images than the smallest classes. This is expected given that in real-life there are certain signs which appear more frequently than others. However, when training the model, I wanted a more uniform distribution so that each class has the same number of training examples and the model, therefore, has an equal number of opportunities to learn each sign. The distribution of training image samples per class is shown in figure 2.

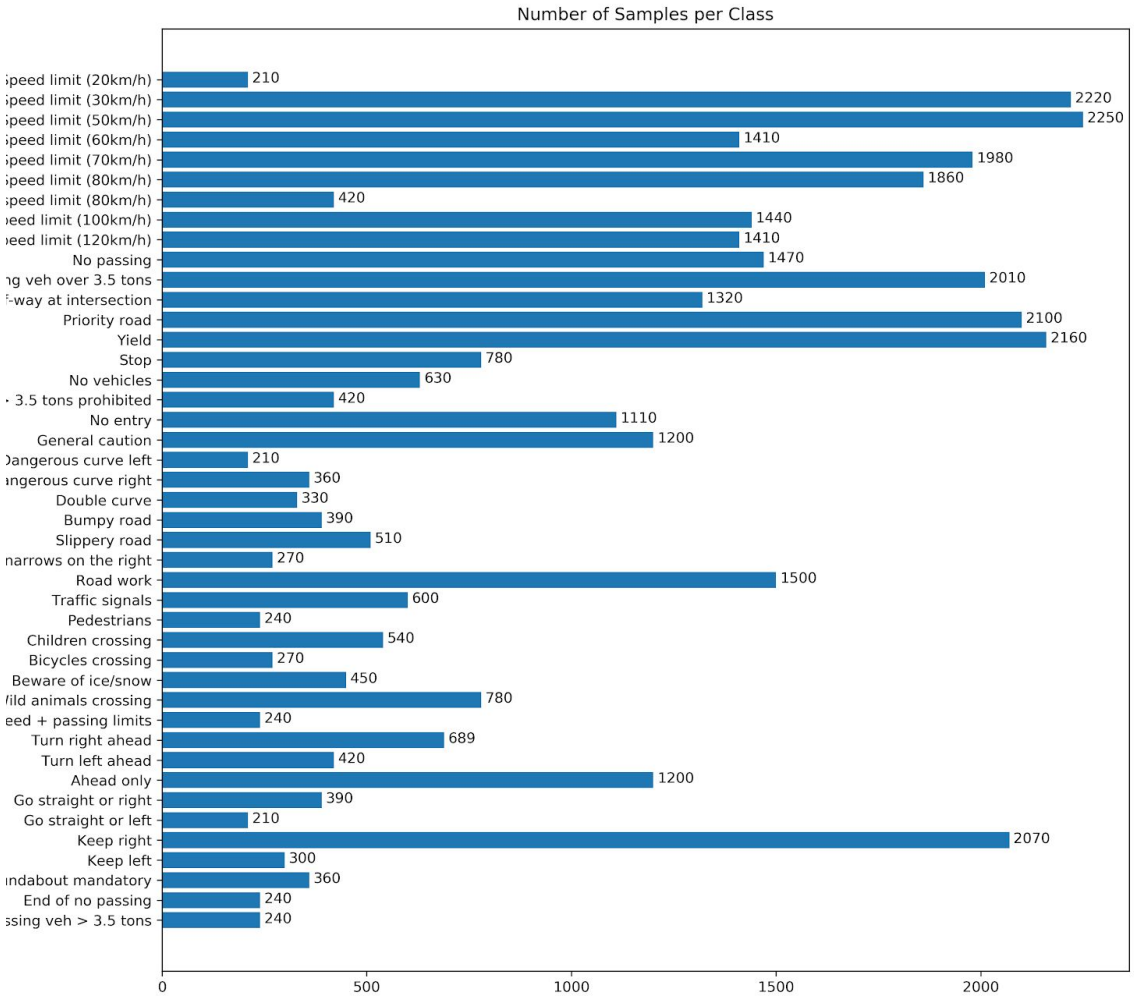

Fig 2. Label Distribution of original training images

Data Preprocessing

Contrast Limited Adaptive Histogram Equalization (CLAHE)

We used this Scikit histogram equalization function, which not only normalizes the images but also enhances local contrast. CLAHE is an algorithm for local contrast enhancement, that uses histograms computed over different tile regions of the image. Local details can therefore be enhanced even in regions that are darker or lighter than most of the images. For color images, the following steps are performed: The image is converted to HSV color space, The CLAHE algorithm is run on the V (Value) channel, The image is converted back to RGB space and returned. For RGBA images, the original alpha channel is removed. This approach enhances an image with low contrast, using a method called histogram equalization, which “spreads out the most frequent intensity values” in an image. The equalized image has a roughly linear cumulative distribution function as shown in figure 3.

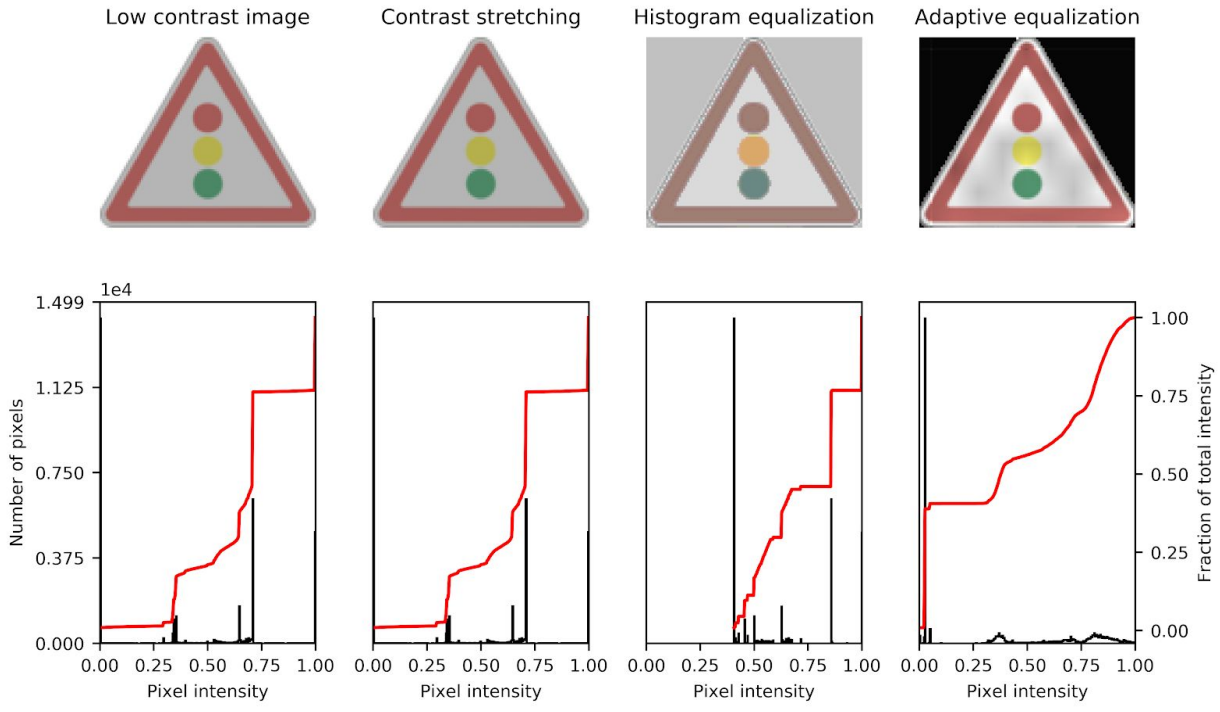

Fig 3. Adaptive Histogram Equalization

Image Augmentation

The image augmentation is done to increase the number of images so that the model has more training examples to learn from and also create the uniform distribution of training images per class. For time being, we have 3k images per class but we may increase the number of images per class during the model training phase to increase the accuracy of the model.

ImageDataGenerator class generates batches of tensor image data with real-time data augmentation. The data will be looped over (in batches). Affine transformations like rotation, shift, shearing, zoom, etc, ZCA whitening are used to generate batches of tensor image data.

```
In [29]: # Transformations applied to RBG training images
         datagen = keras.preprocessing.image.ImageDataGenerator(
             zca_whitening=True,
             rotation_range=10,
             width_shift_range=0.1,
             height_shift_range=0.1,
             shear_range=0.15,
             zoom_range=0.15,
             channel_shift_range=0.1,
             fill_mode='nearest',
             horizontal_flip=False,
             vertical_flip=False
             )
```

Color transformation strategy used:

1. **Color Channel Shifts**:

   This was done to create slight color derivations to help prevent the model from overfitting on specific color shades. This intuitively seemed like a better strategy than grayscaling.

2. **Grayscaling**:

   Converting an image with RGB channels into an image with a single grayscale channel. The value of each grayscale pixel is calculated as the weighted sum of the corresponding red, green and blue pixels as:

$$Y = 0.2125\ R + 0.7154\ G + 0.0721\ B$$

The grayscaled training images sample is shown in figure 4.

Design of upcoming experiments:

We tested a few varieties of models and are planning to apply a convolutional neural network for the task of traffic sign classification. ConvNets are biologically-inspired, multilayer feed-forward architecture that can learn multiple stages of invariant features using a combination of supervised and unsupervised learning. Convolutional neural networks, also known as *convnets*, a type of deep-learning model almost universally used in computer vision applications. The dataset provided by the GTSRB competition presents a number of difficult challenges due to real-world variabilities such as viewpoint variations, lighting conditions (saturation, low-contrast), motion-blur, occlusions, sun glare, physical damage, colors fading, graffiti, stickers and an input resolution as low as 15x15.

All images are down-sampled or up-sampled to 32x32 (dataset samples sizes vary from 15x15 to 250x250) in the data preprocessing phase Importantly, a convnet takes as input tensors of shape **(image_height, image_width, image_channels)** (not including the batch dimension). In this case, we'll configure the convnet to process inputs of size **(32, 32, 3)** for augmented training samples and **(32, 32, 1)** for grayscaled training samples and see the validation accuracy of the model to see if the grayscaled training samples perform well or normalized sample with channel color shift performs well. Finally, we will build the GUI

for uploading an image using Tkinter and classify the uploaded traffic sign using the model trained on training samples.

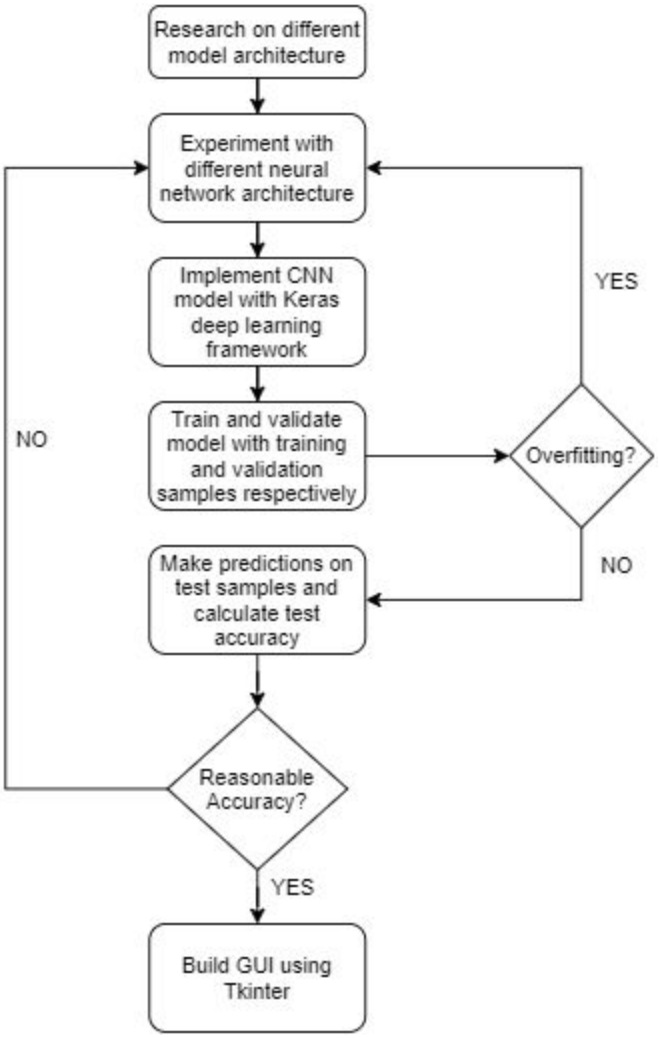

Fig 5. Design of upcoming experiments

Plan for upcoming activities

Name	Work
Junth Basnet	1. Experiment with different network architecture 2. Final model architecture design and visualization 3. Implementing the CNN model using Keras deep learning framework
Sandip Dulal	1. Research 2. Training and validating model
Abin Sainju	1. Research 2. Testing model 3. Design Traffic Sign Classifier GUI using Tkinter

Fig 1. Original Training Image Samples

Fig 5. Normalized (Histogram equalization)Training Image Samples

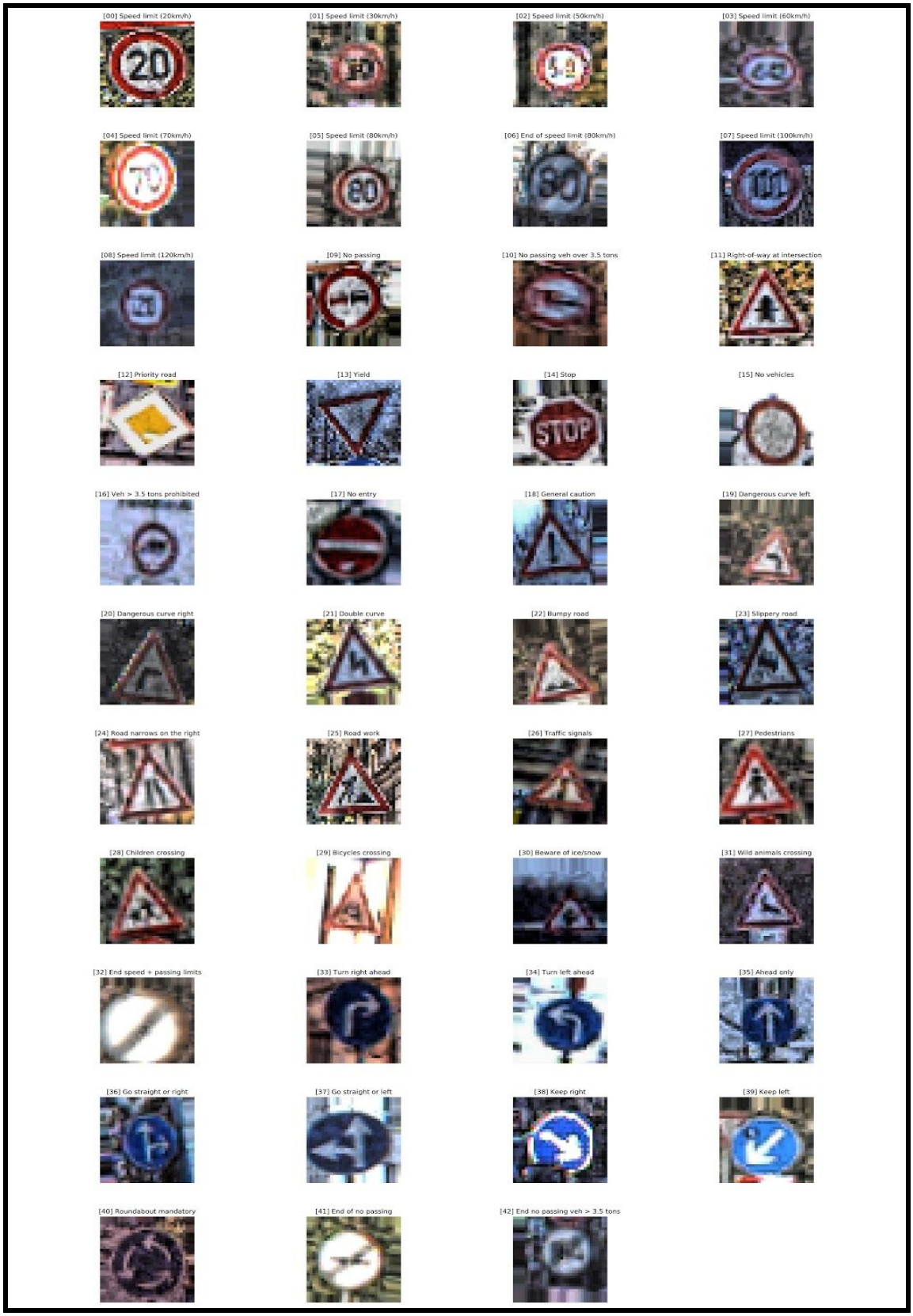

Fig 6. Augmented Training Image Samples

Fig 4. Grayscaled Training Image Samples