Traffic Sign Classification

Ajay Dahiya

C0783939@mylambton.ca

Angelica Serrano C0785370@mylambton.ca

Jonatas Aguiar C0790419@mylambton.ca

Akshita Khatri C0785493@mylambton.ca

Kuldeep Singh C0790419@mylambton.ca

AML-3104 Neural Networks and Deep Learning

Instructor: Mohammad Islam

Lambton College – Toronto

# METHOD

## Data description

The dataset used in this project is the GTSRB (German Traffic Sign Recognition Benchmark) dataset originally from INI Benchmark website. It was a collection of real-life images of German traffic road signs and was formerly used for a competition at IJCNN in 2011.

**Train and Test data**

A total of 3,3799 images were assigned for training data, labeled with 43 classes. Test data consist of 12,630 non-labeled images and 4,410 images were used for validation.

A screenshot of a computer screen

Description automatically generated with medium confidence

***Figure 1 –*** *Preview of GTSRB dataset*

The behavior of train dataset is deemed imbalanced with higher number of images at certain class. For instance, Class 3: Speed limit (50 km/h) has the highest image count while Class 0: Speed limit 20km/h has the lowest value count. Thus, it is more likely for prediction to return Class 3 among others.

Chart

Description automatically generated

***Figure 2 –*** *Graph showing train dataset distribution*

Nevertheless, the original set was used in the same way as it is. Further in the experiment phase, various approaches were conducted including the inclusion of non-German signs and will be discussed in the next sections of this paper.

## Data Pre-processing

Two methods were performed for processing the train data:

**Image augmentation using Augly**

Augly is a newly released library by Facebook and was primarily created for the purpose of data augmentation. It supports four modalities including image and has been used around several applications that identify fake images.

For the initial approach done in this project, Augly was used to perform arbitrary transformations on the train images. These new images were then added to the original set to create input data for the CNN model. The following options were used for augmentation:

1. Sharpen – to increase the sharpness of image
2. Saturate – to increase the intensity of image colors
3. Shuffle – to shuffle the pixels
4. Rotate – to randomly rotate the image to a certain degree
5. Pixelization – to pixelate the image
6. Blur – to decrease the sharpness of image
7. Perspective transform – to change the perspective
8. Change Aspect Ratio - to randomly change the dimension

**Image augmentation using Keras Image Data Generator**

Image Data Generator from Keras is another augmentation library that was utilized in the second approach for CNN modelling. Although the transformation options are quite similar with Augly, this preprocessing library worked a little bit different.

The generated batches of image and labels were transformed into a .flow data in which subsequently used as input images for the Keras model.

Text

Description automatically generated with medium confidence

***Figure 3 –*** *Snippet of .flow data generation. The defined batch size means that the train images are divided into groups of 15 images to be transformed.*

The following geometric transforms were applied on the train set:

1. zoom\_range – to randomly zoom images
2. shear\_range - to randomly apply shearing transformations
3. rotation\_range - to randomly rotate pictures.
4. width\_shift\_range – to randomly transform image horizontally.
5. height\_shift\_range – to randomly transform image vertically.

**OpenCV Image transformation**

As additional image transformation approach, these traditional Opencv pre-processing techniques were utilized as well:

1. Conversion to grayscale
2. Histogram equalization for contrast adjustment.

**Image Normalization**

Images, same with text and numerical data, needs to be scaled or normalized as well. This process works by converting the images into numpy array, diving the values to 255 and producing an output ranging from 0 to 1.

## Convolutional Neural Networks

For this project, three methods were used to produce a CNN model with highest accuracy percentage.

REF:

<https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator>

https://benchmark.ini.rub.de/