Traffic Sign Recognition using CNN

Data Science Capstone (COMP4449)

Aakarsh Sagar

Introduction

Image Classification holds an essential role in the property of **perception** self-driving or automated vehicles. Deep learning techniques, Convolutional Neural Networks, in particular, are best suited for this purpose. A self-driving vehicle needs to see, recognize, and classify objects in its surroundings immediately. The Camera provides the vision, enabling multiple tasks such as classification and segmentation. Other sensors on the vehicle such as LiDAR and RADAR would help with the localization task, i.e., identifying and precisely locating specific objects within the video feed. This project will be focused on Image Classification of traffic signs.

Objectives

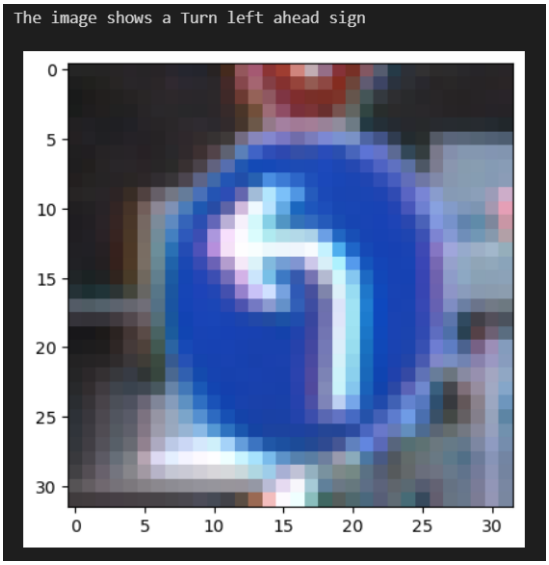
The objective of this project is to build a model using CNN architecture capable of determining the type of traffic sign that is displayed in an image captured under different real-life conditions and showing obstructions, poor lighting, or even the sign being far away from the camera.

Dataset Description

The data is an open dataset from Kaggle. It contains three python pickle files(train, test, validation), all containing thousands of RGB images of traffic signs with noise, different lighting, and different camera angles. There is also a csv file that contains 43 traffic sign names and their respective label numbers.

All the datasets are dictionaries that have a total of 51,839 images with 4 features per image:

['coords', 'labels', 'features', 'sizes']




```
{0: 'Speed limit (20km/h)',
 1: 'Speed limit (30km/h)',
 2: 'Speed limit (50km/h)',
 3: 'Speed limit (60km/h)',
 4: 'Speed limit (70km/h)',
 5: 'Speed limit (80km/h)',
 6: 'End of speed limit (80km/h)',
 7: 'Speed limit (100km/h)',
 8: 'Speed limit (120km/h)',
 9: 'No passing',
 10: 'No passing for vehicles over 3.5 metric tons',
 11: 'Right-of-way at the next intersection',
 12: 'Priority road',
 13: 'Yield',
 14: 'Stop',
 15: 'No vehicles',
 16: 'Vehicles over 3.5 metric tons prohibited',
 17: 'No entry',
```

Figure 1: Sample image from Train data

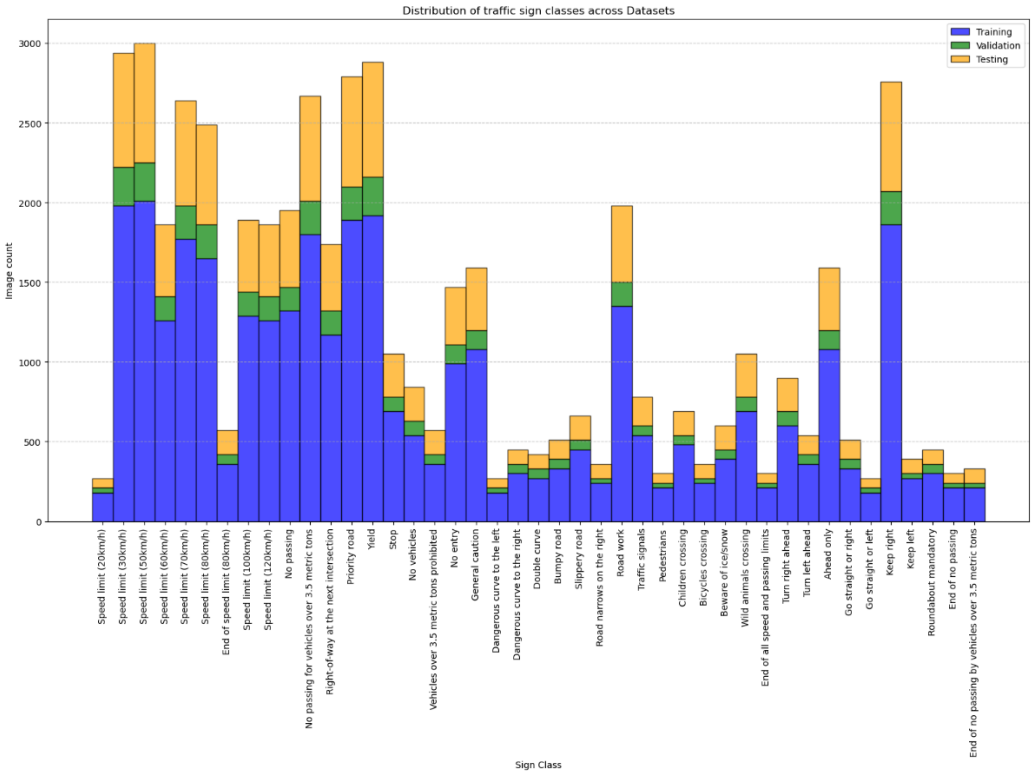
Data Preparation and Analysis

The 'feature' column has the 2D image data. It was resized to 32x32 which would help in the uniformity of sizes across the datasets. The 'labels' feature on the datasets were one-hot encoded. The extracted feature datasets were further converted to NumPy arrays for better compilation and fit on the model.

Final shape of the feature datasets and the label datasets:

Shape of x_train array: (34799, 32, 32, 3)
Shape of x_valid array: (4410, 32, 32, 3)
Shape of x_test array: (12630, 32, 32, 3)
Shape of y_train array: (34799, 43)
Shape of y_valid array: (4410, 43)
Shape of y_test array: (12630, 43)

1. ## Visualization:



A histogram was plot on all three datasets to show the distribution of the images across the 43 labels. It is observed that there are certain labels that have close to 2100 samples while a majority of the labels barely have 500 samples. This non-uniform distribution would affect a model's performance due to lack of enough training data.

Model Implementation

1. 3 Block CNN:

Each CNN block consists of one convolutional layer(Conv2d) that works by applying a filter to the input data to identify features, i.e., feature detector , one MaxPoolingLayer filter that uses an aggregation function to send the pixel with the maximum value to populate the output array. It has a BatchNormalization layer and a dropout layer to prevent overfitting.
There are 2 Dense Layers with 'reLU' activation functions and an output layer with 43 neurons to match the number of classes we need to label.

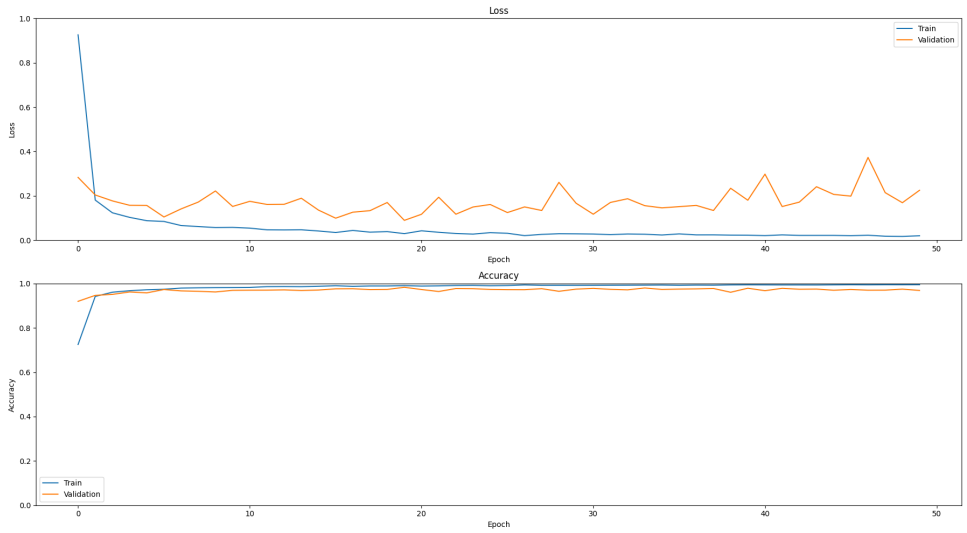


Figure 2 Training & Validation: Accuracy and loss

The model has good training & validation accuracy readings, but the validation loss **does not stabilize** and increases over the epochs. Additional convolutional layers can be added to this model along with regularizations such as learning rate, early stopping, but a pre-trained model such as ResNet, or VGG would help generalize the model better.

2. Pre-trained Model: ResNet50V2 (Imagenet weights, all frozen layers)

A 50 layer pre-trained Model, ResNet was implemented for the training data. A fully connected Dense layer of 512 neurons with reLU activation was connected to the output of the pre-trained model. A Dense layer with number of neurons equal to the number of classes with a softmax activation was added as the final output layer. A learning rate scheduler was implemented with minimum learning as ($1*e^{-4}$), monitored on validation loss. If the loss did not improve after 3 epochs, then the learning rate would decrease.

The model performed worse than before with validation loss increasing every epoch. The validation accuracy got worse after 4 epochs with the final epoch accuracy at 37.44%

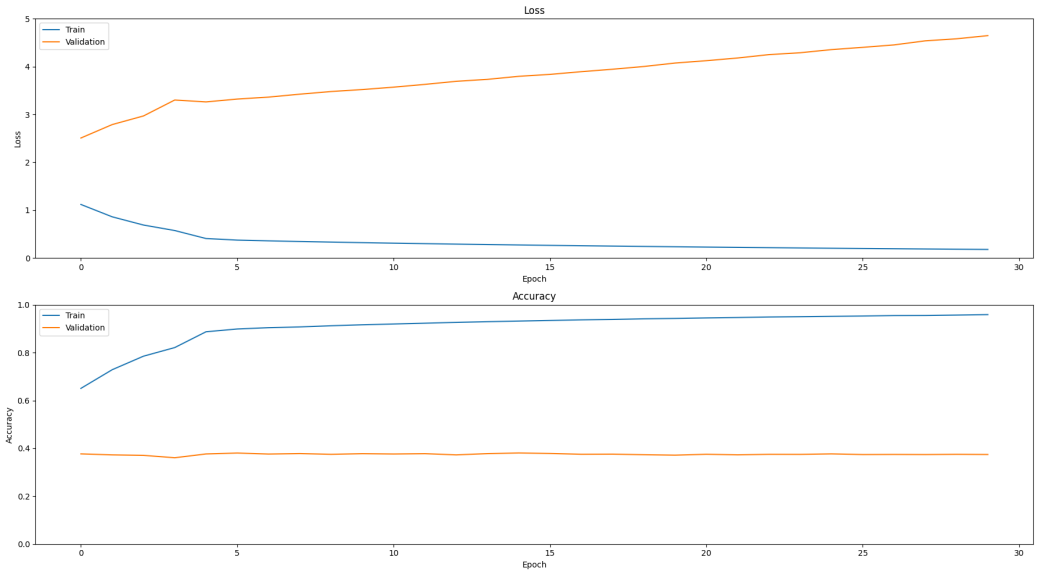


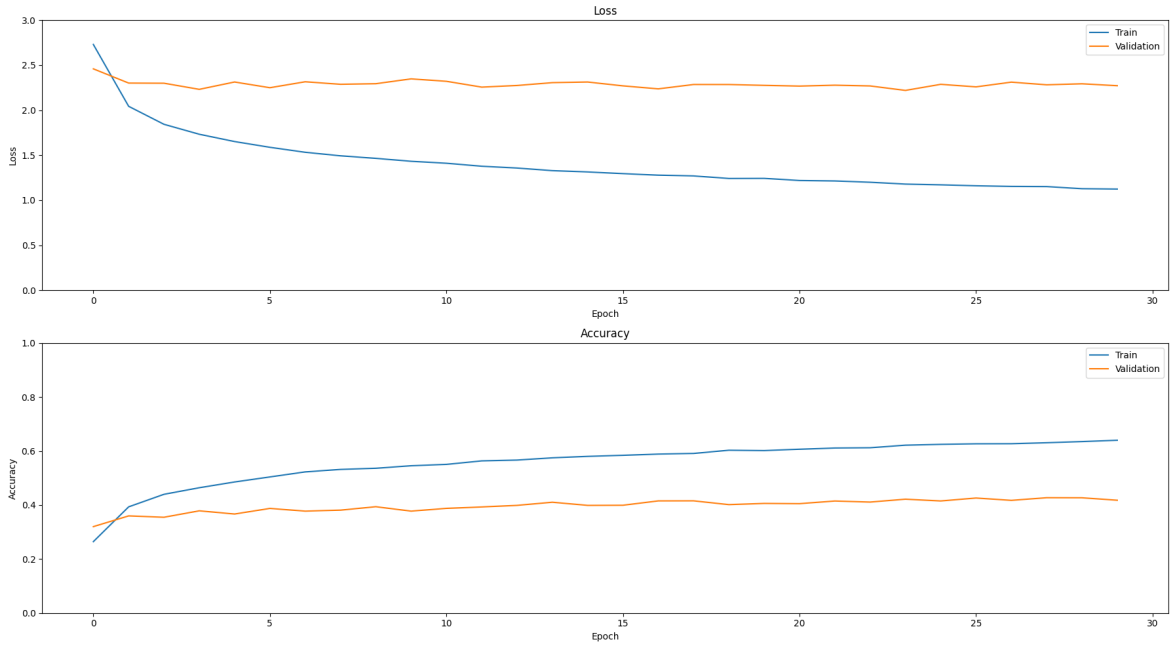
Figure 3 Resnet model 1 accuracy and loss(train&validate)

3. Pre-trained Model: ResNet50V2 (Imagenet weights, final 10 layers frozen, image augmentation)

Image augmentation was performed on the training data to check for improvements in the training data. Augmentations include rotation, mirroring, zooming, brightness, contrast tuning.

Another learning rate scheduler was implemented to the model.

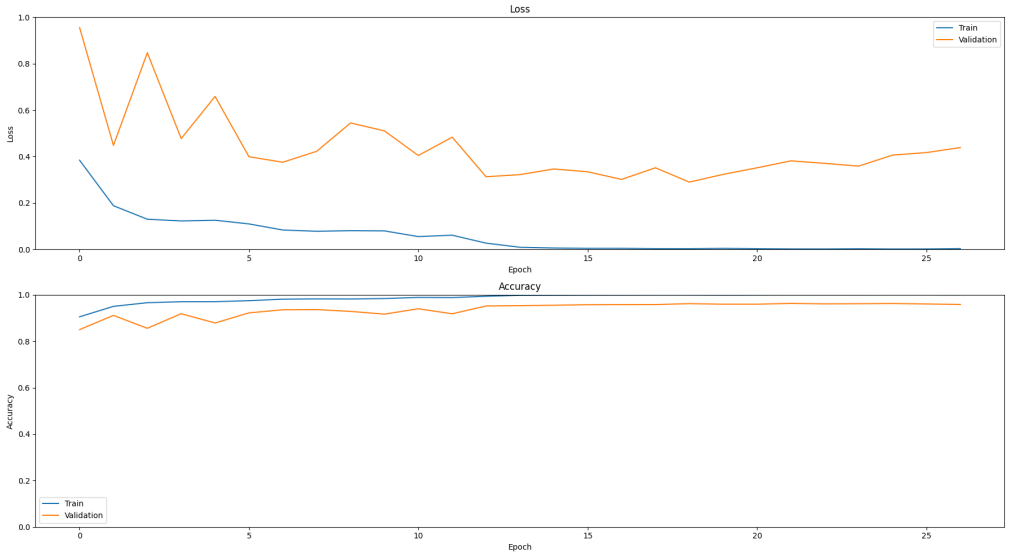
output: The model performance was bad. The training and validation accuracy did not improve over 60%, and the loss was high at 1.1 and 2.72.



4. Pre-trained Model: ResNet50V2 (No weights, all layers trainable, no image augmentation)

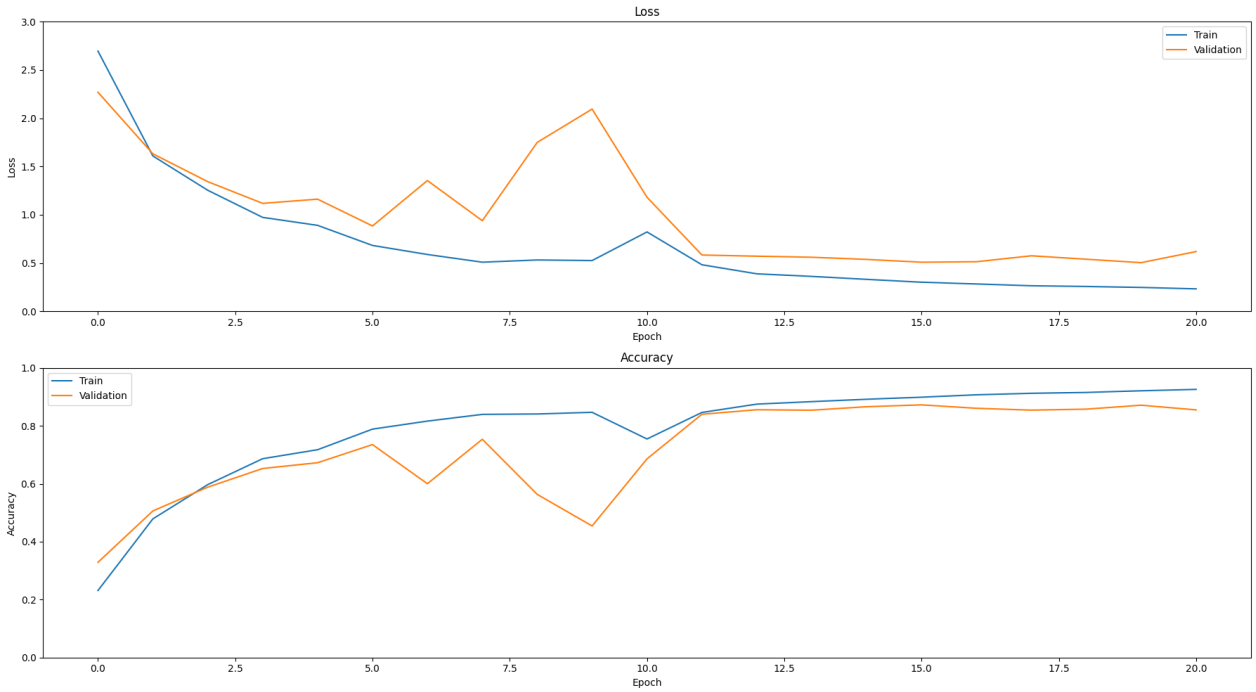
To check whether the model improves, the weights were removed, and all layers were trainable. Two fully connected dense layers were added in addition to the final output layer. Regularizations such as ModelCheckPoint, EarlyStopping, and Learning rate scheduler were implemented to prevent overfitting.

The model still overfit the training data though it performed well in terms of training and validation accuracy. The validation loss still did not stabilize towards the end. Image augmentation would be required as another regularization criteria to prevent overfitting.

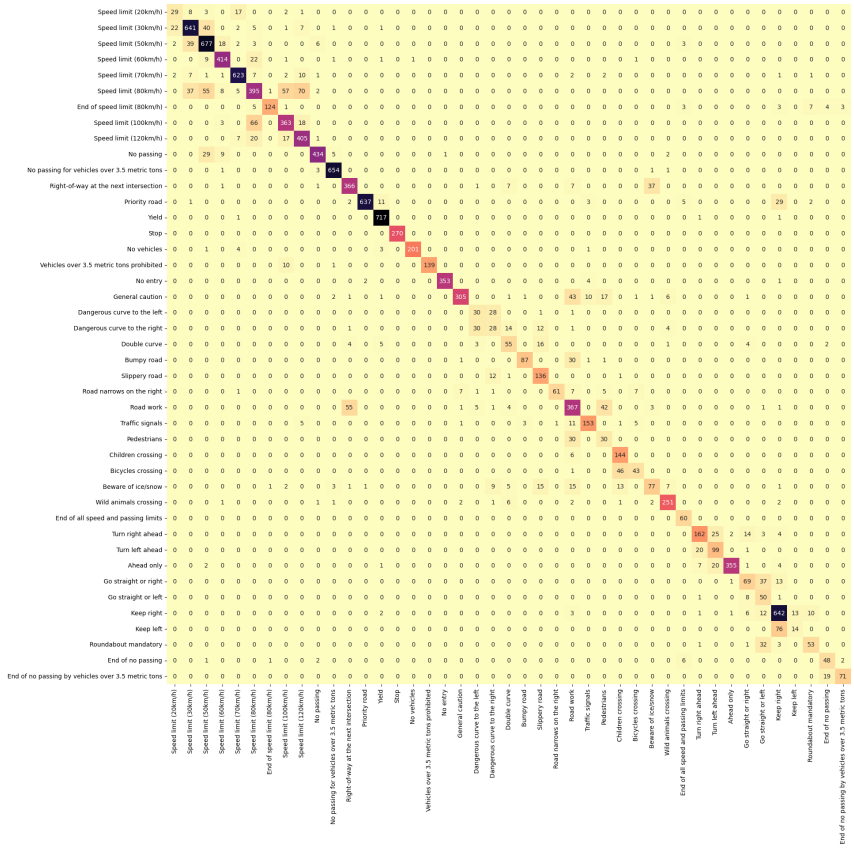


5. Pre-trained Model: ResNet50V2 (No weights, all layers trainable, image augmentation)

Image augmentation such as rotation, mirroring was performed on the training dataset as an added regularization parameter to the previous model.
The model performed well with the validation loss and training loss stabilizing towards the end.
The training and validation accuracy was 92.68% and 85.5% respectively.



The model performance on the test dataset was good at 86%.



There were some incorrect predictions. This can be improved with better data handling, stricter regularizations.

	Class_Label	Accuracy	Predicted_Label	Count
0	Speed limit (20km/h)	48.33%	Speed limit (30km/h)	8
1	Speed limit (20km/h)	48.33%	Speed limit (50km/h)	3
2	Speed limit (20km/h)	48.33%	Speed limit (70km/h)	17
3	Speed limit (20km/h)	48.33%	Speed limit (100km/h)	2
4	Speed limit (20km/h)	48.33%	Speed limit (120km/h)	1
5	Speed limit (30km/h)	89.03%	Speed limit (20km/h)	22
6	Speed limit (30km/h)	89.03%	Speed limit (50km/h)	40
7	Speed limit (30km/h)	89.03%	Speed limit (70km/h)	2
8	Speed limit (30km/h)	89.03%	Speed limit (80km/h)	5
9	Speed limit (30km/h)	89.03%	Speed limit (100km/h)	1
10	Speed limit (30km/h)	89.03%	Speed limit (120km/h)	7
11	Speed limit (30km/h)	89.03%	No passing for vehicles over 3.5 metric tons	1
12	Speed limit (30km/h)	89.03%	Yield	1
13	Speed limit (50km/h)	90.27%	Speed limit (20km/h)	2
14	Speed limit (50km/h)	90.27%	Speed limit (30km/h)	39
15	Speed limit (50km/h)	90.27%	Speed limit (60km/h)	18