CLASSIFICATION OF TRAFFIC-SIGN AND TRAFFIC-LIGHT USING DEEP LEARNING ALGORITHMS

15CS496L MAJOR PROJECT

Submitted by

RA1611003010791 MEDHA SINGH

RA1611003010863 GONA SRAVANTHI

Guided by

Dr. T.K.Sivakumar (100487, SRMIST)

In partial fulfilment of the requirements for the degree of

BACHELOR OF TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

February 2020

SRM
INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

BONAFIDE CERTIFICATE

Certified to be the project report titled **"CLASSIFICATION OF TRAFFIC-SIGN AND TRAFFIC-LIGHT USING DEEP LEARNING ALGORITHMS "** is the bonafide work done by **MEDHA SINGH** (Reg.No:RA1611003010791) and **GONA SRAVANTHI** (Reg No:RA1611003010863), of CSE B.Tech. Degree course in the theory **15CS496L MAJOR PROJECT** during the academic year 2019-20.

Sign of the Guide

Name: Dr. T.K. Sivakumar (100487)

Department: CSE

Date of Submission: 8th February, 2020

ABSTRACT

Image-based traffic-sign and light detection, classification, and recognition are one of the important fundamentals for the intelligent transport systems. Extensive research has shown that good performance can be achieved on public data sets by different state-of-the-art approaches, especially the deep learning methods (CNN using tensor-flow and keras). Nonetheless, deep learning methods depend on extensive computing resources. Also, these approaches mostly fixate on single image detection and recognition task, which is not applicable in real-world applications. A computational framework for traffic sign and light detection, classification, and recognition task will be explored.

TABLE OF CONTENTS

INTRODUCTION

About, History and General Terms

Traffic sign detection and recognition based on computer vision and pattern recognition is being exanimated for several purposes, such as autonomous driving and assisted driving. Recognition of traffic signs allows a driver to be alert about inappropriate actions and potentially alarming situations. There are two errands in a typical traffic sign recognition system: catching the locations and sizes of traffic signs in dynamic scene images (traffic sign detection) and classifying the detected traffic signs into their particular sub-classes (traffic sign classification).

There are a lot of researchers working on this challenging task with the already prominent or specially designed vision algorithms. However, it is not easy to compare these methods since there did not exist a publicly available data set until the release of the German Traffic Sign Recognition Benchmark (GTSRB) and German Traffic Sign Detection Benchmark (GTSDB) in 2011 and 2013 respectively. Since then, analysts can calculate and compare their algorithms on the same benchmarks.

The traffic light recognition problem has been extensively examined. The conventional approaches can be broadly divided into three sub-divisions: template matching, circular extraction, and color distribution. The first category applies the template of red and green lights to the extracted region. In the second category, Hough Transform is applied to detect the circular shape from the image. The third category is mainly color segmentation. Sensitivity to the lighting conditions is a major disadvantage of these methods.

Though the structures and presentation of traffic signs are different across the world, the distinct color and shape characteristics of traffic signs provide important clues to design detection methods. Shape or edge detection methods are popular in the detection literature. Different shape detection methods are constructed to detect circle, triangle or octagon. Shape and edge detection methods can also be used to extract the accurate position of a traffic sign.

In recent years, with the development of machine learning methods especially deep learning methodologies, the machine learning based detection methods have progressively become the main stream algorithms. There are three main traffic sign detection structures: AdaBoost based detection, Support Vector Machine (SVM) based detection, and Neural Networks (NN) based detection. These detection architectures have many derivatives with different input features, different training methods or different detection processes. The machine learning based detection methods have achieved the-state-of-the-art results in some aspects. This paper provides a critical review of traffic sign detection and offer suggestions for future research areas in this challenging problem domain.

LITERATURE SURVEY

Survey

Ref No.	Objective	Algorithm	Dataset	Performance Metrics	Result	Remark
1	Towards Real-Time Traffic Sign Detection and Classification	Colour probability model, SVM and CNN	Chinese Traffic Sign Dataset (CTSD)	real-time traffic sign recognition system	Detects and classifies traffic signs at~6 fps on 1360×800 image	Method could be accelerated with GPU, which could further improve the computationalefficiency.
2	Traffic Light Recognition With High Dynamic Range Imaging and Deep Learning	• non-parametric multicolour saliency model • CNN • temporal trajectory tracking	large dual-channel dataset	Problem caused by lighting conditions and rear lights or head lights of the vehicles ahead	Performance of the proposed traffic light recognition is better than the state-of-the-art deep learning object detector which uses only bright images.	Future investigation will look into using both dark and bright images as input to the CNN network
3	Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network	Convolutional neural network (CNN)	▪ German Traffic Sign Detection Benchmark (GTSDB) ▪ ImageNet dataset ▪ San Diego Traffic Sign	Boundary estimation	Detector can run with framerate of 7 FPS on low-power mobile platforms.	The proposed method can be applied not only to traffic sign but also to any other planar objects having standard shapes
4	Classification of Traffic Signs: The European Dataset	Convolutional neural network (CNN)	Real-world European dataset	real-time traffic sign recognition system in Europe.	Accuracies of 99.37% and 98.99% were the best results obtained for training the GTSRB and our European dataset respectively	In future work intents to take into account the class imbalance problem to improve recognition accuracy
5	Accurate and Reliable Detection of Traffic Lights Using Multiclass Learning and Multi-object Tracking	Color extraction and candidates selection followed by processing by PCANet and SVM, finally, tracking and forecasting techniques are applied	Build their own data set of traffic lights from recorded driving videos, including circular lights and arrow lights in different directions.Data set is available at http://computing.wpi.edu/Dataset.html.	Detect multiple types of green and red traffic lights accurately and reliably	The proposed system can successfully detect the traffic lights on the scene with high accuracy and stable results	▪ Different light condition, color distortion, motion blur and variance of scenes may compromise the system performance in the real world ▪ the proposed method is not expected to work at night

Table 1

Ref No.	Objective	Algorithm	Dataset	Performance Metrics	Result	Remark
6	Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey	▪ HOG features used with a support vector machine (SVM) ▪ use a cascaded classifier trained with some type of boosting ▪ neural networks and genetic algorithms	▪ German TSR Benchmark (GTSRB) ▪ KULBelgiumTrafficSignsDataset(KULDataset) ▪ Swedish Traffic Signs Data set (STS Data set) ▪ RUG Traffic Sign Image Database (RUG Data set) ▪ Stereopolis Database	real-time traffic sign recognition system	This paper has provided an overview of the state of sign detection	Currently, every new approach presented uses a new data set for testing, making comparisons between papers hard
7	Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network	Convolutional Neural Network (CNN)	▪ German Traffic Sign Recognition Benchmark(GTSRB) ▪ German Traffic Sign Detection Benchmark(GTSDB)	Process all categories, including both symbol-based and text-based traffic signs	System gets the state-of-the-art result	Increasing the speed of the system and post-processing with geometry information are considered part of future work
8	MR-CNN: A Multi-Scale Region-Based Convolutional Neural Network for Small Traffic Sign Recognition	Multiscale region-based convolutional neural network(MR-CNN).	Tsinghua Tencent 100K	▪ Detecting small traffic signs ▪ improving the image resolution and semantic information for small traffic sign detection	Obtains state-of-the-art results (an F1-measure of 86.0% for small (sizes∈ (0,32] pixels), 93.5% for medium (sizes∈ (32,96] pixels), and 90.1% for large (sizes∈ (96,200] pixels)	In future work, plan to focus on designing an efficient training algorithm to differentiate hard negative samples from easy positive samples
[9]	Traffic Light Recognition With High Dynamic Range Imaging and Deep Learning.	1. Non-parametric multicolour saliency model. 2. CNN. 3. Temporal trajectory tracking.	Large dual-channel dataset.	Problem caused by lighting conditions and rear lights or head lights of the vehicles ahead.	Performance of the proposed traffic light recognition is better than the state-of-the-art deep learning object detector which uses only bright images.	Future investigation will look into using both dark and bright images as input to the CNN network
[10]	Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network.	Convolutional neural network (CNN).	1. German Traffic Sign Detection Benchmark (GTSDB). 2. ImageNet dataset. 3. San Diego Traffic Sign.	Boundary estimation.	Detector can run with framerate of 7 FPS on low-power mobile platforms.	The proposed method can be applied not only to traffic sign but also to any other planar objects having standard shapes.

Table 2

Ref. No.	Objective	Algorithm used	Datasets or Input Parameters	Performance metrics used	results	remark
[11]	Classification of Traffic Signs: The European Dataset.	Convolutional neural network (CNN).	Real-world European dataset.	real-time traffic sign recognition system in Europe.	Accuracies of 99.37% and 98.99% were the best results obtained for training the GTSRB and our European dataset respectively.	In future work intents to take into account the class imbalance problem to improve recognition accuracy.
[12]	Accurate and Reliable Detection of Traffic Lights Using Multiclass Learning and Multi-object Tracking.	Color extraction and candidates selection followed by processing by PCANet and SVM, finally, tracking and forecasting techniques are applied.	Build their own data set of traffic lights from recorded driving videos, including circular lights and arrow lights in different directions.Data set is available at http://computing. wpi. edu/Data set.html.	Detect multiple types of green and red traffic lights accurately and reliably.	The proposed system can successfully detect the traffic lights on the scene with high accuracy and stable results.	1. Different light condition, color distortion, motion blur and variance of scenes may compromise the system performance in the real world. 2. The proposed method is not expected to work at night.
[13]	Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey.	1. HOG features used with a support vector machine (SVM). 2. Use a cascaded classifier trained with some type of boosting. 3. Neural networks and genetic algorithms.	1. German TSR Benchmark (GTSRB). 2. KUL Belgium Traffic Signs Dataset (KULDataset). 3. Swedish Traffic Signs Data set (STS Data set). 4. RUG Traffic Sign Image Database (RUG Data set). 5. Stereopolis Database.	Real-time traffic sign recognition system.	This paper has provided an overview of the state of sign detection.	Currently, every new approach presented uses a new data set for testing, making comparisons between papers hard.
[14]	Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network.	Convolutional Neural Network (CNN).	1. German Traffic Sign Recognition Benchmark(GTSRB). 2. German Traffic Sign Detection Benchmark(GTSDB).	Process all categories, including both symbol-based and text-based traffic signs.	System gets the state-of-the-art result.	Increasing the speed of the system and post-processing with geometry information are considered part of future work.
[15]	MR-CNN: A Multi-Scale Region-Based Convolutional Neural Network for Small Traffic Sign Recognition.	Multiscale region-based convolutional neural network(MR-CNN).	Tsinghua Tencent 100K.	1. Detecting small traffic signs. 2. improving the image resolution and semantic information for small traffic sign detection.	Obtains state-of-the-art results (an F1-measure of 86.0% for small (sizes∈ (0,32] pixels), 93.5% for medium (sizes∈ (32,96] pixels), and 90.1% for large (sizes∈ (96,200] pixels).	In future work, plan to focus on designing an efficient training algorithm to differentiate hard negative samples from easy positive samples.

Table 3

Inference from the survey

This literature survey reflects on the various systems and methods employed in traffic sign and traffic light detection, recognition and classification and their subsequent refinements.

Traffic sign detection methods can be divided into five categories: color based methods, shape-based methods, color and shape-based methods, machine learning based methods, and LIDAR based methods.

The earliest system in this literature survey proposed a TSR system in high resolution images with high accuracy, and the detection and recognition were partly robust to color changes, blurred edges and partial occlusions. To address the problem of multiclass traffic sign detection in high-resolution images with high speed and accuracy, the system developed a ROI extraction method called HCRE and the SFC-tree detector for rapidly detecting different types of traffic signs in a dynamic environment. The proposed HCRE method was able to reduce the detection regions of the SFC-tree detector to save detection time of the TSR system. The traffic sign recognition method based on the ESRC utilizes a content dictionary and an occlusion dictionary to sparsely represent traffic signs, which were able to reach occlusion robust recognition with compact dictionaries. The results of the validation experiments showed that the method was fast and robust to color changes.

Later a purely vision-based traffic light detection method was introduced, which was robust to different illuminations. Most of the previous approaches concerning such problem used color thresholding solely or used simple template matching methods, thus they were affected by light changes. System proposed a new traffic light candidate extraction algorithm, AdaBSF, which was suitable and effective in that task. The experimental results suggested that the proposed method was fast and robust. By applying proposed method on collected video data and public dataset, improvement over conventional approaches was achieved. The whole detection procedure was performed in real time.

Next a system proposed a common detection framework for detecting three important classes of objects in traffic scenes. The proposed framework introduced spatially pooled features as a part of aggregated channel features to enhance the feature robustness and employed detectors of three important classes to detect multiple objects. The detection speed of the framework was fast since dense features needed only to be evaluated once rather than individually for each detector.

Down the stream another system proposed two fast algorithms for traffic sign detection and classification, respectively. The detection module first extracted traffic sign proposals by using color probability model and MSER region detector. Then an SVM classifier was used to filter out false positives and classify the remaining proposals into their super classes based on a novel color HOG feature. The classification module further classified the detected signs into their specific sub-classes within each super class.

Following this a survey presented an overview of the current state of traffic light recognition (TLR) research in relation to driver assistance systems. The approaches from the surveyed paper were broken down into choices made for color space, detection, features, classification, and tracking. For color space, there did not exist a clear tendency towards one in particular. Most detection approaches relied on color or shape for finding TL candidate’s other relied on spot light detection in a single intensity channel. BLOB analysis was generally used to remove bad TL candidates, this was done based on prior knowledge of the properties of TL BLOBs. Furthermore, some of the best performing approaches used detailed maps of the route and temporal information to improve performance. Many papers utilized manually specified models of TLs, which consisted of color, shape, and structural features, to do state detection of TL candidates. Other used trained features such as HoG, LBP, and 2D Gabor wavelets, classified using SVM. A few relyed on template matching or neural networks using the color and/or shape. The tracking stage was dominated by temporal filtering, while more advanced approaches include HMM, IMM, and CAMSHIFT.

Next, a system that could detect multiple types of green and red traffic lights accurately and reliably was proposed for traffic light detection. Color extraction and blob detection were applied to locate the candidates with proper optimization. A classification and validation method using PCANet was then used for frameby-frame detection. Multiobject tracking method and forecasting technique were employed to improve accuracy and produce stable results. As an additional contribution, it also built a traffic light data set from the videos captured by a camera mounted behind the windshield.

In another paper, a unified framework for traffic sign detection, recognition and tracking in videos has been studied. The videos were recorded using a vehicle mounted camera. The central idea of the framework was that a pre-trained off-line trained detector could be improved by an on-line updated detector, which is synchronous to a local predictor based on a Kalman filter. The framework has three aspects: The first aspect is, decreasing the false positive detections by involving the spatial distribution priori knowledge. The second aspect is adopting an on-line strategy for incremental tracking which takes the motion model (Kalman Filter) and the appearance model (on-line detector) into consideration simultaneously. The final aspect being, adoption of a scale-based fusion algorithm to make the final result more accurate.

Subsequently, a new data-driven system was proposed to recognize all categories of traffic signs in low quality short videos that were captured by a camera mounted on a car. The ROIs of traffic signs were first extracted using MESRs on multichannel images. A new multi-task CNN structure to refine and classify the ROIs in a uniform framework was proposed. Synthetic images and street view data were used to generate larger number of data with low cost to train a reliable network. Fusion of the recognition outputs of all frames was used to get final result for a video.

Later research reflected on the efficient traffic sign detection method where locations of traffic signs are estimated together with their precise boundaries. The object bounding box detection problem was generalised and an object pose estimation problem was formulated, and the problem was effectively modeled using CNN based on the recent advances in object detection networks. The estimated pose of traffic signs was used to transform the boundary of traffic sign templates into the input image plane, providing precise boundaries with high accuracy. To achieve practical detection speed, the best-performing base networks were explored and unnecessary layers were pruned out of the network by considering the characteristics of traffic signs.

Next, a highly compact deep convolutional neural network called MicronNet was introduced for real-time embedded traffic sign recognition. By designing a highly optimized network architecture, each layer's microarchitecture was optimized to have as few parameters as possible, along with macroarchitecture augmentation and parameter precision optimization, the resulting MicronNet network achieved a good balance between model size, accuracy and inference speed. The resulting MicronNet possessed a model size of just ~510,000 parameters (~27x fewer parameters than state-of-the-art) and ~1 MB, it required just ~10 million multiply-accumulate operations to perform inference (with 32.19 ms of time-to-compute on a Cortex-A53 high efficiency processor), while still achieving a accuracy of 98.9% on GTSRB dataset(German traffic sign recognition benchmark), thus achieving human-level performance. These experimental results showed that very small yet accurate deep neural network architectures could be designed for real-time traffic sign recognition that are well-suited for embedded scenarios.

Following this, a traffic sign European dataset was proposed, which deals with intra-class variability from 6 countries (France, Belgium, Germany, Croatia, Netherlands and Sweden). Such characteristic is a crucial aspect when driving from one country to another for autonomous vehicles, since a classifier does not perform properly when traffic signs (pictographs or text) slightly differ from each other. This is a vital issue in Europe, considering that countries are relatively close to each other. For this reason, a traffic sign dataset that contemplates the aforementioned problem was proposed to make an important contribution for intelligent vehicles.

On similar ground, a small traffic sign detection framework named MR-CNN that employs a two-stage fusion strategy was proposed. The MR-CNN framework integrates multiple levels of convolution feature and contextual information. At the detection stage, the fused feature map with sufficient information was used to generate the region proposals. The fusion strategy of different convolution layers was designed by using deconvolution, compression and normalization. At the classification stage, a fused feature for the fully connected layer was constructed and the multi-scale contextual regions were used to exploit the surrounding information for a given object proposal. The fused feature map was focused on improving the resolution and semantic information of small traffic signs, while the fused feature provided more discriminative representations of the contextual regions.

In yet another research paper, the traffic sign detection methods have been studied and divided into five categories: shape-based methods, color based methods, color and shape-based methods, machine learning based methods, and LIDAR based methods.

Down the line, a real-time traffic light recognition system based on HDR imaging and deep learning was proposed. Different from the state-of-the-art approaches which use bright images only, the low and high exposures were used in this system for traffic light detection and recognition, respectively. The clear background of the low exposure (dark) channel made the traffic light detection much more reliable than the existing color-image-based approaches. Also, the candidate regions in the high exposure corresponding to the dark channel could be recognized well because of the rich texture. By using region of interest and salience map, which can prune the traffic light candidates efficiently, the number of the traffic light candidates to be verified by CNN were reduced significantly. This made the system fast and robust to noise (such as vehicle's rear lights). Furthermore, a tracking technology was developed to further improve the accuracy and reliability. The experimental results, which are based on a large database collected from real roads, had shown that the performance of the proposed traffic light recognition was better than the state-of-the-art deep learning object detector which uses only bright images. The candidate selection from the dark images allowed to prevent false positives caused by the traffic signs, pedestrian and temporary road signs which had similar color with traffic light.

NOVELTY IN METHODOLOGY

Keras framework for traffic sign detection, classification, and recognition task has been explored.

We would introduce traffic light detection and recognition to the model already developed so that the system can now actively detect and identify traffic lights and traffic signs with improved accuracy.

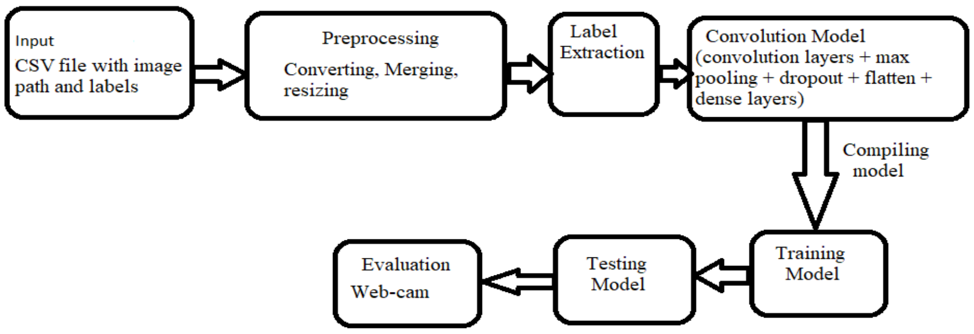

Input
CSV file with image path and labels
Preprocessing
Converting, Merging, resizing
Label Extraction
Convolution Model (convolution layers + max pooling + dropout + flatten + dense layers)
Compiling model
Training Model
Testing Model
Evaluation
Web-cam

Figure 1

MODULES

1. Dataset

The traffic-sign dataset was obtained from GTSRB (German Traffic Sign Recognition Benchmark). It is an image classification dataset. The images are photos of traffic signs. The images are classified into 43 classes. The training set contains 39209 labeled images and the test set contains 12630 images.

Structure

The training set archive is structures as follows:

- One directory per class
- Each directory contains one CSV file with annotations ("GT-<ClassID>.csv") and the training images
- Training images are grouped by tracks
- Each track contains 30 images of one single physical traffic sign

Annotation format

Annotations are provided in CSV files. Fields are separated by “;"(semicolon) for e.g. Figure 1, Figure 2. Annotations contain the following information:

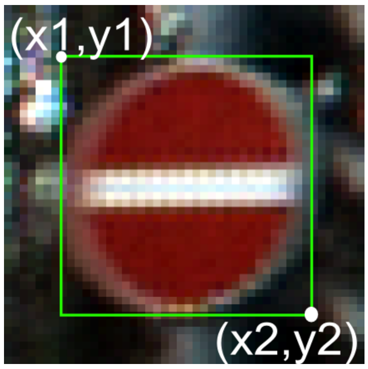
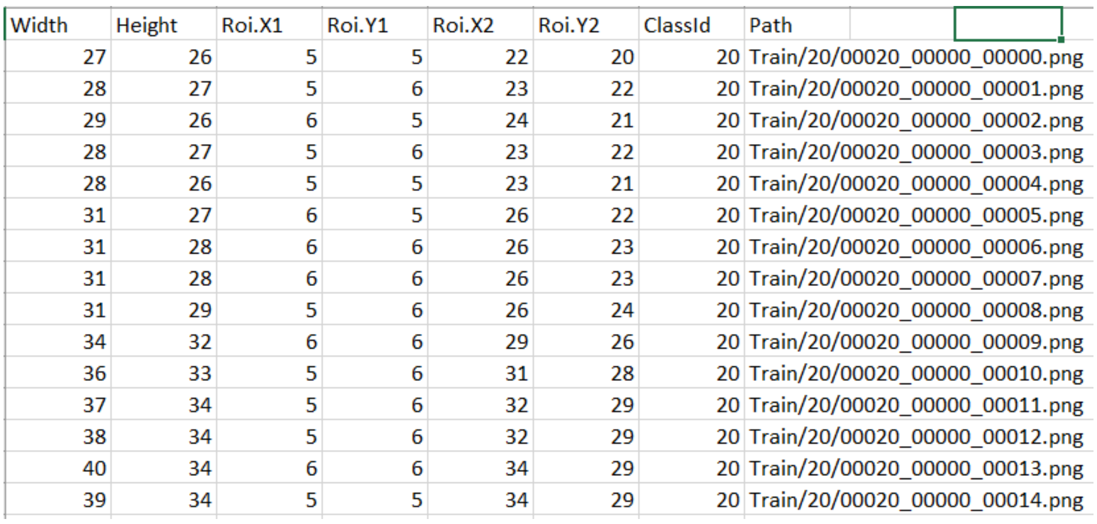


Figure 2

- Path: Path of corresponding image
- Width: Width of the image
- Height: Height of the image

- ROI.x1: X-coordinate of top-left corner of traffic sign bounding box
- ROI.y1: Y-coordinate of top-left corner of traffic sign bounding box
- ROI.x2: X-coordinate of bottom-right corner of traffic sign bounding box
- ROI.y2: Y-coordinate of bottom-right corner of traffic sign bounding box

  The training data annotations will additionally contain
- ClassId: Assigned class label



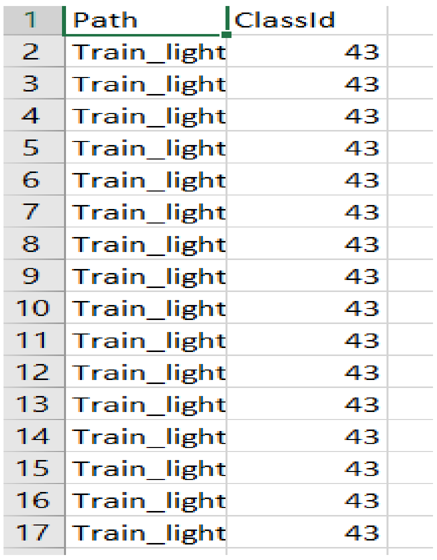
Width	Height	Roi.X1	Roi.Y1	Roi.X2	Roi.Y2	ClassId	Path
27	26	5	5	22	20	20	Train/20/00020_00000_00000.png
28	27	5	6	23	22	20	Train/20/00020_00000_00001.png
29	26	6	5	24	21	20	Train/20/00020_00000_00002.png
28	27	5	6	23	22	20	Train/20/00020_00000_00003.png
28	26	5	5	23	21	20	Train/20/00020_00000_00004.png
31	27	6	5	26	22	20	Train/20/00020_00000_00005.png
31	28	6	6	26	23	20	Train/20/00020_00000_00006.png
31	28	6	6	26	23	20	Train/20/00020_00000_00007.png
31	29	5	6	26	24	20	Train/20/00020_00000_00008.png
34	32	6	6	29	26	20	Train/20/00020_00000_00009.png
36	33	5	6	31	28	20	Train/20/00020_00000_00010.png
37	34	5	6	32	29	20	Train/20/00020_00000_00011.png
38	34	5	6	32	29	20	Train/20/00020_00000_00012.png
40	34	6	6	34	29	20	Train/20/00020_00000_00013.png
39	34	5	5	34	29	20	Train/20/00020_00000_00014.png

Figure 3

The traffic-light dataset is obtained from Worcester Polytechnic Institute website from embedded computing laboratory.

Training Data

- The training data are collected in Worcester, MA, USA during summer.
- ROI Samples: Extracted from frames in Frames_GT_wHolder. Can be used for training classifier. Figure 3 shows CSV folder of the same.

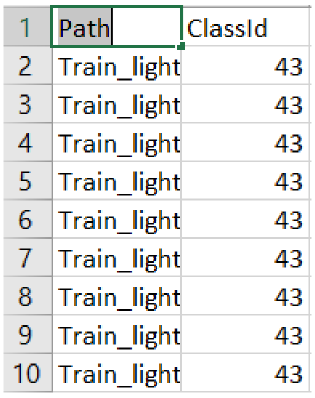


1	Path	ClassId
2	Train_light	43
3	Train_light	43
4	Train_light	43
5	Train_light	43
6	Train_light	43
7	Train_light	43
8	Train_light	43
9	Train_light	43
10	Train_light	43
11	Train_light	43
12	Train_light	43
13	Train_light	43
14	Train_light	43
15	Train_light	43
16	Train_light	43
17	Train_light	43

Figure 4

Both of the above datasets are combined into a single data frame, which consists of two parameters, which are path of the images and their labels.

Figure 4 shows CSV file for traffic-light.



1	Path	ClassId
2	Train_light	43
3	Train_light	43
4	Train_light	43
5	Train_light	43
6	Train_light	43
7	Train_light	43
8	Train_light	43
9	Train_light	43
10	Train_light	43

Figure 5

Below in Figure 5 is the distribution of the final dataset. It has 124653 images along with total of 63 labels.

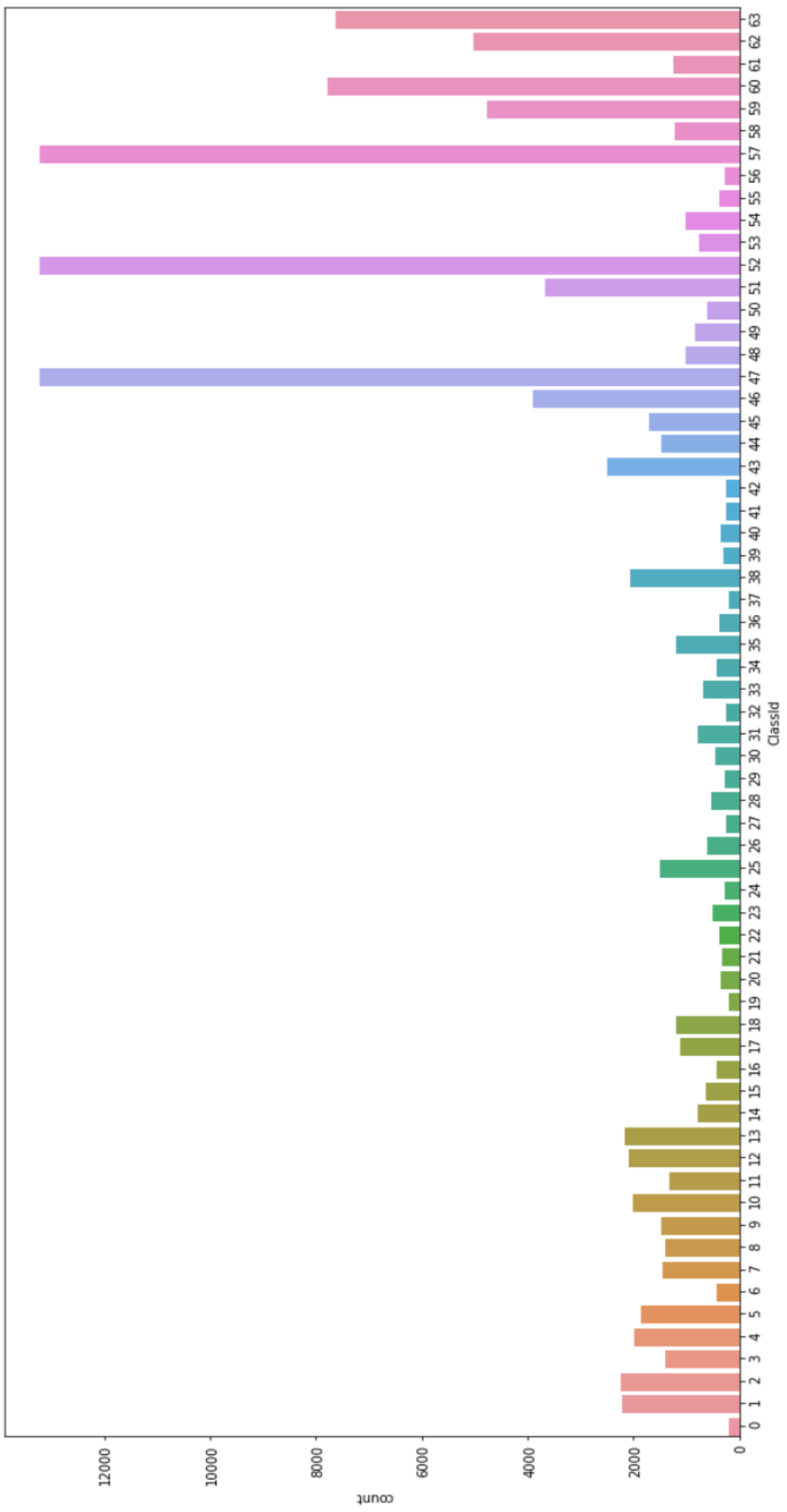

count
0
2000
4000
6000
8000
10000
12000
ClassId
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63

Figure 6

Issues with data

There are numerous issues with dataset which represent the real-world problems faced in actual traffic sign recognition.

1. **Class imbalance:** Apparently dataset is very unbalanced, some of the classes having 2000 samples and some of them having only 200 samples. This imbalance biases the model to predict classes with higher number of samples more frequently than the ones with a smaller number of samples to achieve better accuracy.
2. **High contrast variation:** The images differ significantly in terms of contrast and brightness. It is difficult to human to understand and classify some of these signs which are in total darkness.
3. **Small dataset:** Even though we have about 35k train images, they are not quite enough to perform well in all general case scenario. More data also helps with overfitting.

2. Pre-processing

- **Resizing and cropping**

The images are not uniform and there is a need to resize and crop them to remove the unnecessary parts and reduce the noise. This can significantly affect the accuracy of the model.

- **Equalization**

High contrast variation among the images calls for contrast normalization. The Contrast-limited adaptive histogram equalization (CLAHE for short) algorithm partitions the images into contextual regions and applies the histogram equalization to each one. This evens out the distribution of used grey values and thus makes hidden features of the image more visible as seen in Figure 6.

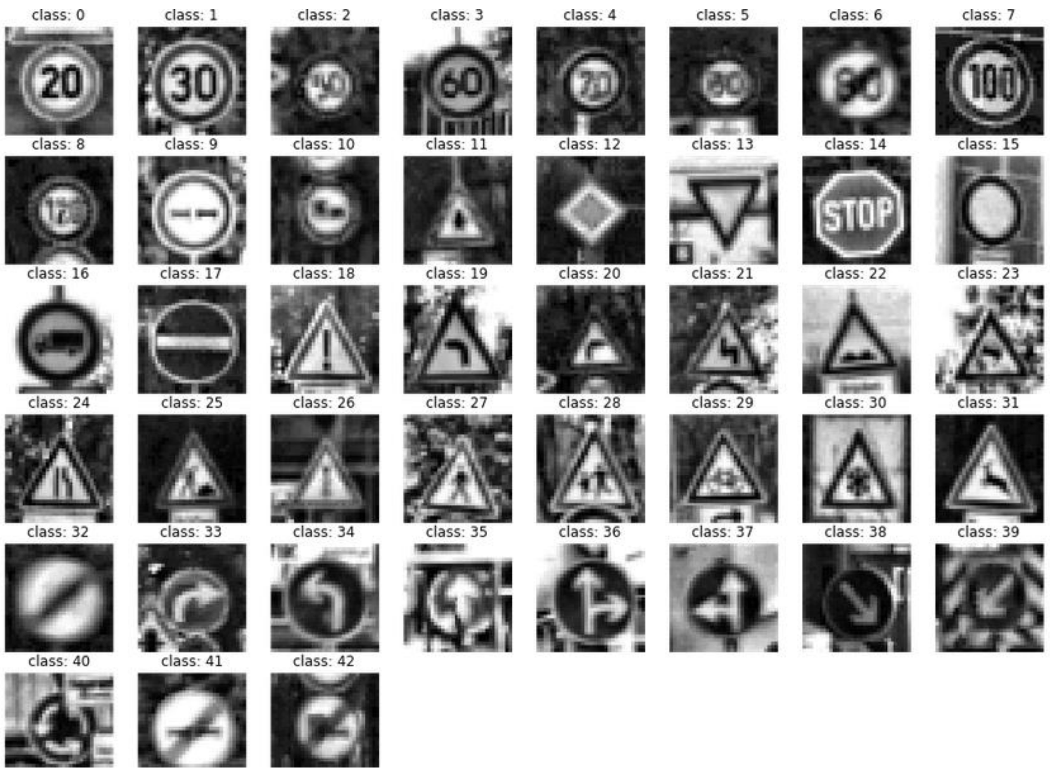


Figure 7

- **Up-sampling/ Down-sampling**

Up-sampling is the process of randomly duplicating observations from the minority class in order to reinforce its signal.

There are several heuristics for doing so, but the most common way is to simply resample with replacement.

First, we'll import the resampling module from Scikit-Learn:

Next, we'll create a new DataFrame with an up-sampled minority class. Here are the steps:

1. First, we'll separate observations from each class into different DataFrames.
2. Next, we'll resample the minority class **with replacement**, setting the number of samples to match that of the majority class.
3. Finally, we'll combine the up-sampled minority class DataFrame with the original majority class DataFrame.

Down-sampling involves randomly removing observations from the majority class to prevent its signal from dominating the learning algorithm.

The most common heuristic for doing so is resampling without replacement.

The process is similar to that of up-sampling. Here are the steps:

1. First, we'll separate observations from each class into different DataFrames.
2. Next, we'll resample the majority class without replacement, setting the number of samples to match that of the minority class.
3. Finally, we'll combine the down-sampled majority class DataFrame with the original minority class DataFrame.

3. Dataset Visualization and Feature Selection

Descriptive Analytics is one of the core components of any analysis life-cycle pertaining to a data science project or even specific research. Data aggregation, summarization and visualization are some of the main pillars supporting this area of data analysis. Since the days of traditional Business Intelligence to even in this age of Artificial Intelligence, Data Visualization has been a powerful tool and has been widely adopted by organizations owing to its effectiveness in abstracting out the right information, understanding and interpreting results clearly and easily.

4. Classification Using Deep Learning Algorithm

Machine Learning is a part of Artificial Intelligence system that aims to create a perfect robot which can do things human can do better by understanding the logic and reasons in a way human can. One Image processing technique related to ML which helps in identification of targets is CNN.

A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an image as input, assign importance to various objects in the image and be able to differentiate one from the other. CNN requires very less pre-processing as compared to other classification algorithms in the bank. Usually the filters are hand-engineered, with enough training, but CNN have the ability to learn these filters with precision and accuracy. The Spatial and Temporal dependencies in an image can be successfully captured by CNN through the application of relevant filters. Due to the reduction in the number of parameters involved and reusability of weights, the architecture performs a better fitting to the image dataset. In other words, the network can be trained to understand the sophistication of the image better as in Figure 7.

We used CNN with 6 convolution layers, 3 max pooling layers, 2 dense layers and 1 flattening layer. We also used 4 dropout layers to prevent overfitting. We used SGD optimizer with ‘nesterov’ and 0.9 momentum to optimize the model. We ran the fitting process for 20 epochs with batch-size 32, validation split 20% and with a learning rate of 0.01.

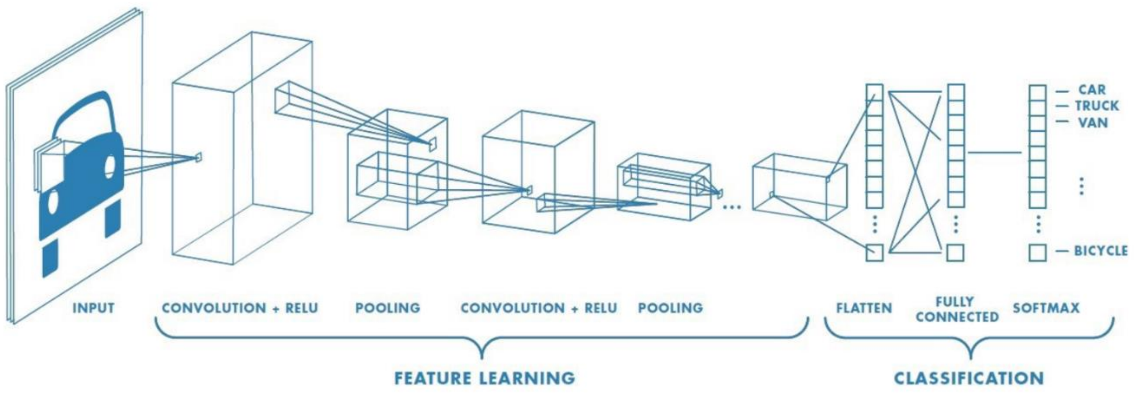


Figure 8

- Convolution Layers

  Input is an RGB image which has been separated by its three-color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc. The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

  The objective of the Convolution Operation is to extract the high-level features such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network, which has the wholesome understanding of images in the dataset.

- Max pooling Layers

  Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling**.**

- Dense Layer

  Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

- Flattening Layer

  After converting the input image into a suitable form for our Multi-Level Perceptron, flattening the image into a column vector should be done. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the Softmax Classification technique.

- Drop-out Layer

Machine learning is ultimately used to predict outcomes given a set of features. Therefore, anything we can do to generalize the performance of our model is seen as a net gain as shown in Figure 8. Dropout is a technique used to prevent a model from overfitting. Dropout works by randomly setting the outgoing edges of hidden units (neurons that make up hidden layers) to 0 at each update of the training phase.

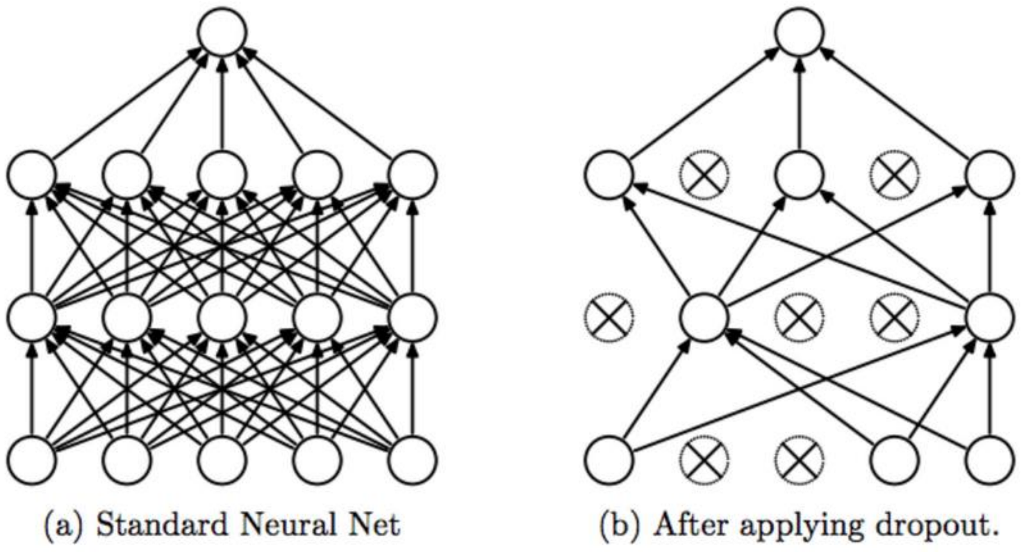


Figure 9

- SGD(Stochastic gradient descent) Optimizer

The most common method to train a neural network is by using gradient descent (SGD). Define a loss function l(θ) that expresses how well the weights & biases θ allow the network to fit the training data. A higher loss means that the network is bad and makes a lot of errors while a low loss generally means that the network performs well. Then train the network by adjusting the network parameters in a way that reduces the loss.

Formally the update rule in SGD is defined like this:

$\theta t+1=\theta t - \eta \nabla l(\theta)$

Here we take the gradient ∇ of the loss function l which tells us in which direction we have to move through parameter space to increase the loss. Then we go in the opposite direction $-\nabla l$ (in which the loss decreases) and move by a distance dependent on the learning rate η. This works very well in most cases and is the foundation of much of modern deep learning.

Momentum is essentially a small change to the SGD parameter update so that movement through the parameter space is averaged over multiple time steps. This is done by introducing a velocity component v. Momentum speeds up movement along directions of strong improvement (loss decrease) and also helps the network avoid local minima. It is intuitively related to the concept of momentum in physics.

With momentum, the SGD update rule is changed to: $vt+1=\mu vt-\eta \nabla l(\theta)\theta t+1=\theta t+vt+1$

Here v is the velocity and μ is the momentum parameter which controls how fast the velocity can change and how much the local gradient influences long term movement. At every time step the velocity is updated according to the local gradient and is then applied to the parameters.

Nesterov momentum is a simple change to normal momentum. Here the gradient term is not computed from the current position θt in parameter space but instead from a position $\theta intermediate=\theta t+\mu vt$. This helps because while the gradient term always points in the right direction, the momentum term may not. If the momentum term points in the wrong direction or overshoots, the gradient can still "go back" and correct it in the same update step.

The revised parameter update rule is:

$vt+1=\mu vt-\eta\nabla l(\theta+\mu vt)\theta t+1=\theta t+vt+1$

- Epochs and Batch-size

One Epoch is when an entire dataset is passed forward and backward through the neural network only once.

Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

It is important to pass the full dataset multiple times to the same neural network. But as we are using a limited dataset and to optimise the learning and the graph, we are using Gradient Descent which is an iterative process. So, updating the weights with single pass or one epoch is not enough.

One epoch leads to underfitting of the curve in the graph (below) in Figure 9.

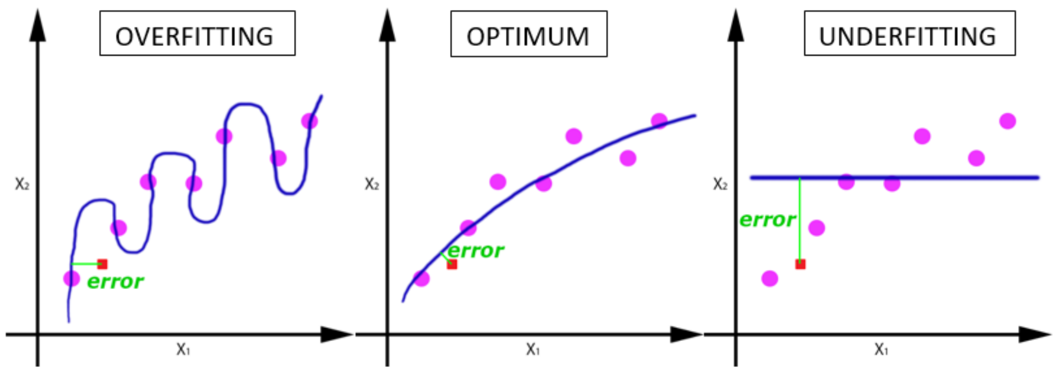


Figure 10

As the number of epochs increases, a greater number of times the weights are changed in the neural network and the curve goes from underfitting to optimal to overfitting curve.

- Learning Rate

  The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated. Choosing the learning rate is challenging as a value too small may result in a long training process that could get stuck, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

  The learning rate may be the most important hyperparameter when configuring your neural network. Therefore, it is vital to know how to investigate the effects of the learning rate on model performance and to build an intuition about the dynamics of the learning rate on model behaviour.

5. Result Visualization, Validation, Testing

The most important step after creating an application is testing if the system works. Validation is the process to validate or check if we are getting the desired output or not. Validation is the most important step in Software Development cycle as once the product is made and ready to deploy, if there are any errors in it, it can be caught in this stage but once the product is deployed without proper validation then things become too hard.

In our project the code is first trained with images and then they are validated. If the system works properly the validation shows it else, we need to fix some bugs in the system.

- For traffic-signal

  loss= 0.06189291878279733

  accuracy= 0.9795775059980104

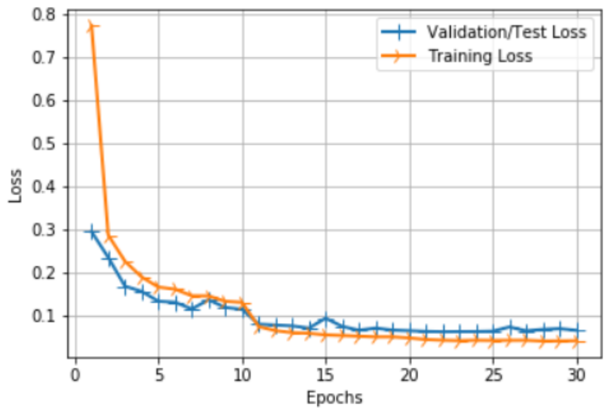


Figure 11

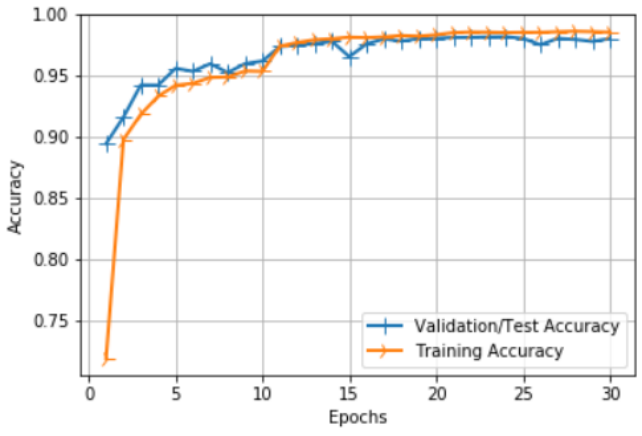


Figure 12

- For traffic-sign

  loss= 0.009647773702439211

  accuracy= 0.9974496301963784

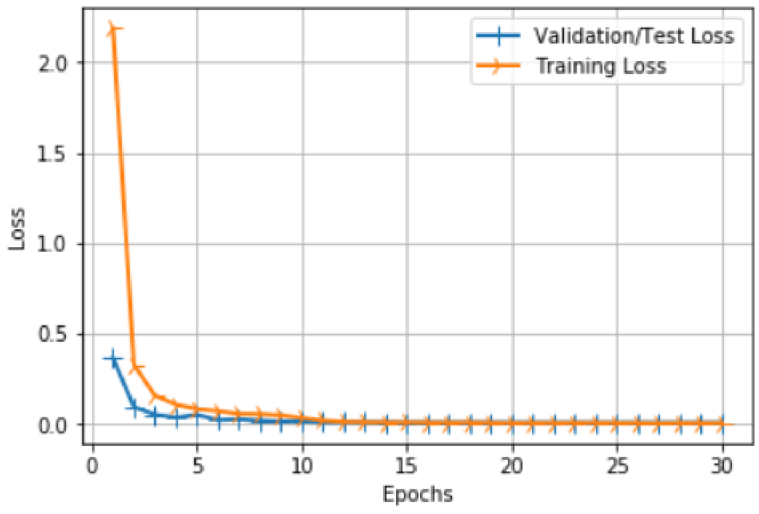


Figure 13

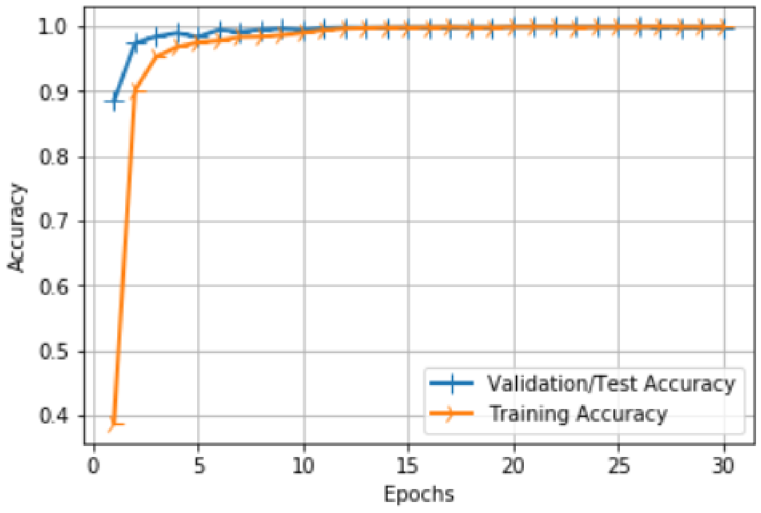


Figure 14

- For combined model of traffic-sign and traffic-light

  loss= 0.08463251031704712

  accuracy= 0.9700856734888148

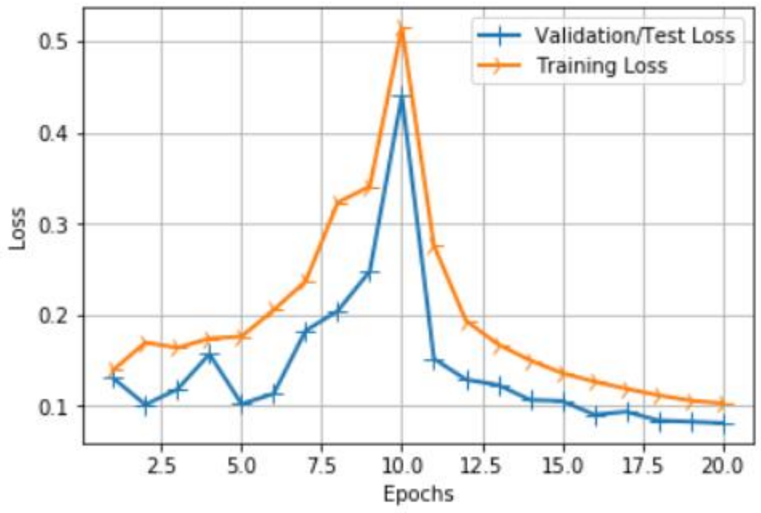


Figure 15

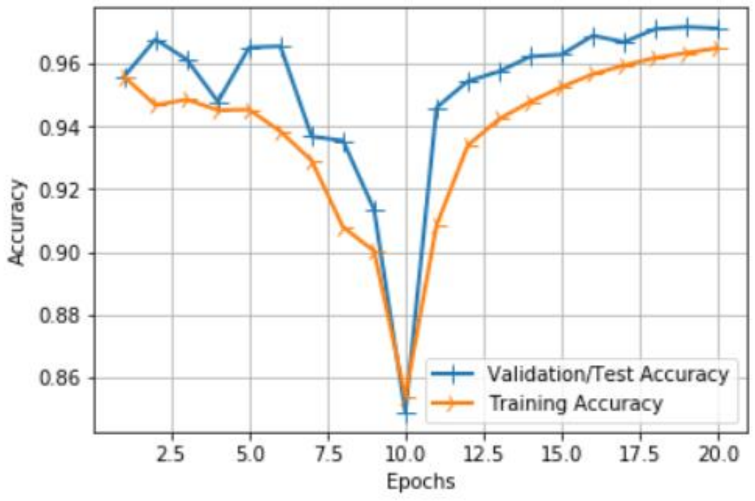


Figure 16

6. Testing Model output using web-camera

To show the output of our project we have used web-camera available on our system to take input image and show the classification of the input image.

CONCLUSION

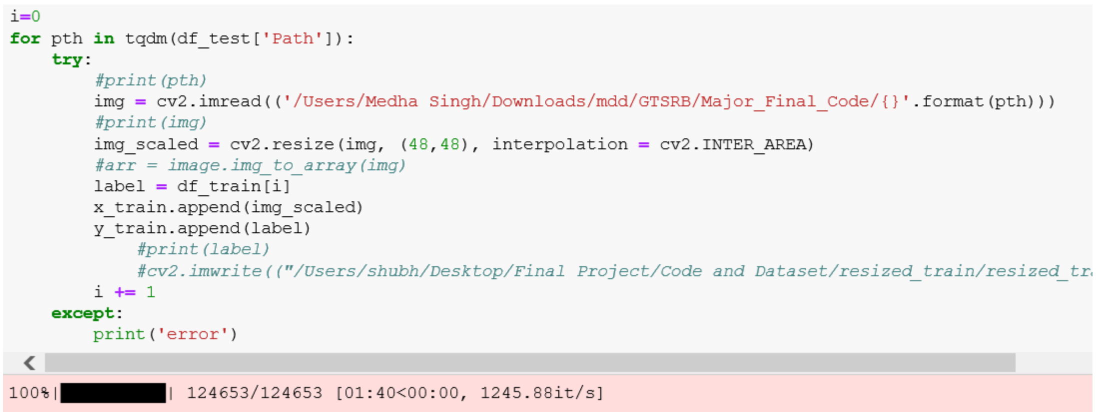
A computational framework for traffic-sign and traffic-signal detection, classification, and recognition task was successfully created.

Traffic light detection and recognition was introduced to the model already developed so that the system can now actively detect and identify traffic lights and traffic signs.

The future work is to study richer features for traffic sign and traffic light detection, and to improve model accuracy.

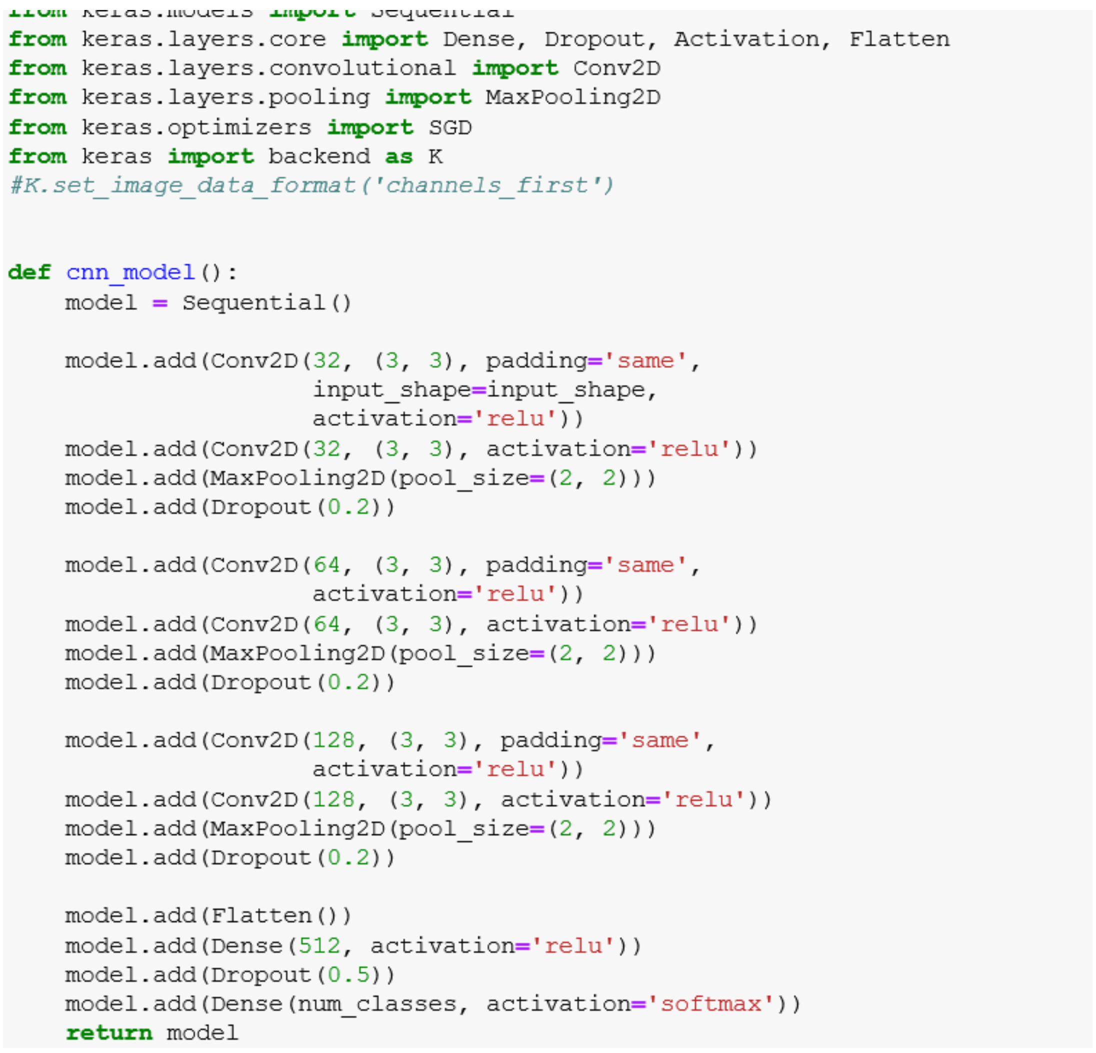
The saliency information or object proposal can also be explored for faster detection.

APPENDIX



```
i=0
for pth in tqdm(df_test['Path']):
    try:
        #print(pth)
        img = cv2.imread(('/Users/Medha Singh/Downloads/mdd/GTSRB/Major_Final_Code/{}'.format(pth)))
        #print(img)
        img_scaled = cv2.resize(img, (48,48), interpolation = cv2.INTER_AREA)
        #arr = image.img_to_array(img)
        label = df_train[i]
        x_train.append(img_scaled)
        y_train.append(label)
            #print(label)
            #cv2.imwrite(("/Users/shubh/Desktop/Final Project/Code and Dataset/resized_train/resized_tr
        i += 1
    except:
        print('error')
```

```
100%|██████████| 124653/124653 [01:40<00:00, 1245.88it/s]
```



```
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation, Flatten
from keras.layers.convolutional import Conv2D
from keras.layers.pooling import MaxPooling2D
from keras.optimizers import SGD
from keras import backend as K
#K.set_image_data_format('channels_first')


def cnn_model():
    model = Sequential()

    model.add(Conv2D(32, (3, 3), padding='same',
                     input_shape=input_shape,
                     activation='relu'))
    model.add(Conv2D(32, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(128, (3, 3), padding='same',
                     activation='relu'))
    model.add(Conv2D(128, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(512, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    return model
```

```
from keras.optimizers import SGD

model = cnn_model()

# let's train the model using SGD + momentum
lr = 0.01
sgd = SGD(lr=lr, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
model.summary()
```

```
import numpy as np
import cv2
import pickle
```

```
frameWidth= 640         # CAMERA RESOLUTION
frameHeight = 480
brightness = 180
threshold = 0.75         # PROBABLITY THRESHOLD
font = cv2.FONT_HERSHEY_SIMPLEX
```

```
# SETUP THE VIDEO CAMERA
cap = cv2.VideoCapture(0)
cap.set(3, frameWidth)
cap.set(4, frameHeight)
cap.set(10, brightness)
# IMPORT THE TRANNIED MODEL
pickle_in=open("model_final_trained.p","rb")  ## rb = READ BYTE
model=pickle.load(pickle_in)
```

```
def grayscale(img):
    img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    return img
def equalize(img):
    img =cv2.equalizeHist(img)
    return img
def preprocessing(img):
    img = grayscale(img)
    img = equalize(img)
    img = img/255
    return img
def getCalssName(classNo):
    if   classNo == 0: return 'Speed Limit 20 km/h'
    elif classNo == 1: return 'Speed Limit 30 km/h'
    elif classNo == 2: return 'Speed Limit 50 km/h'
    elif classNo == 3: return 'Speed Limit 60 km/h'
    elif classNo == 4: return 'Speed Limit 70 km/h'
    elif classNo == 5: return 'Speed Limit 80 km/h'
    elif classNo == 6: return 'End of Speed Limit 80 km/h'
    elif classNo == 7: return 'Speed Limit 100 km/h'
    elif classNo == 8: return 'Speed Limit 120 km/h'
```

REFRENCES

- Yuan Yuan, Zhitong Xiong, and Qi Wang, “An Incremental Framework for Video-Based Traffic Sign Detection, Tracking, and Recognition ”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 18, NO. 7, JULY 2017
- Hengliang Luo, Yi Yang, Bei Tong, Fuchao Wu, and Bin Fan, “Traffic Sign Recognition Using a Multi-Task Convolutional Neural Network”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 19, NO. 4, APRIL 2018
- Zhilu Chen and Xinming Huang , “Accurate and Reliable Detection of Traffic Lights Using Multiclass Learning and Multi-object Tracking”, IEEE Intelligent Transportation Systems MagazineYear, Vol. 8, NO. 4 , 2016
- Qichang Hu, Sakrapee Paisitkriangkrai, Chunhua Shen, Anton van den Hengel, and Fatih Porikli, “Fast Detection of Multiple Objects in Traffic Scenes With a Common Detection Framework”,IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 4, APRIL 2016
- Yi Yang, Hengliang Luo, Huarong Xu, and Fuchao Wu, “Towards Real-Time Traffic Sign Detection and Classification ”, IEEE TRANSACTIONS ON INTELLIGENTTRANSPORTATIONSYSTEMS, VOL. 17, NO. 7, JULY 2016
- ChunshengLiu, Faliang Chang, Zhenxue Chen, and Dongmei Liu, “Fast Traffic Sign Recognition via High-Contrast Region Extraction and Extended Sparse Representation ”,IEEE TRANSACTIONS ON

INTELLIGENTTRANSPORTATIONSYSTEMS, VOL. 17, NO. 1, JANUARY 2016

- Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund,and Mohan Manubhai Trivedi, “Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 17, NO. 7, JULY 2016
- Zhenwei Shi, Zhengxia Zou, and Changshui Zhang, “Real-Time Traffic Light Detection With Adaptive Background Suppression Filter ”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS , VOL. 17 , NO. 3 , MARCH 2016
- Jian-Gang Wang ,and Lu-Bing Zhou, “Traffic Light Recognition With High Dynamic Range Imaging and Deep Learning”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 20, NO. 4, APRIL 2019
- Hee Seok Lee and Kang Kim, “Simultaneous Traffic Sign Detection and Boundary Estimation Using Convolutional Neural Network ”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 19, NO. 5, MAY 2018
- Alexander Wong , Mohammad Javad Shafiee , and Michael St. Jules, “MicronNet:A Highly Compact Deep Convolutional Neural Network Architecture for Real-Time Embedded Traffic Sign Classification”, IEEE ACCESS, JOURNAL ARTICLE, VOL. 6, 2018
- Citlalli Gamez Serna and Yassine Ruichek, “Classification of Traffic Signs: The European Dataset”, IEEE ACCESS, JOURNAL ARTICLE, VOL. 6, 2018

- Zhigang Liu , Juan Du , Feng Tian and Jiazheng Wen, “MR-CNN: A Multi-Scale Region-Based Convolutional Neural Network for Small Traffic Sign Recognition”, IEEE ACCESS, JOURNAL ARTICLE, VOL. 7, 2019
- Andreas Møgelmose, Mohan Manubhai Trivedi, and Thomas B. Moeslund, “Vision-Based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey ”, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, VOL. 13, NO. 4, DECEMBER2012
- Chunsheng Liu , Shuang Li , Faliang Chang and Yinhai Wang, “Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives”, IEEE ACCESS, JOURNAL ARTICLE, VOL. 7, 2019