Machine Learning Engineer Nanodegree

Capstone Proposal

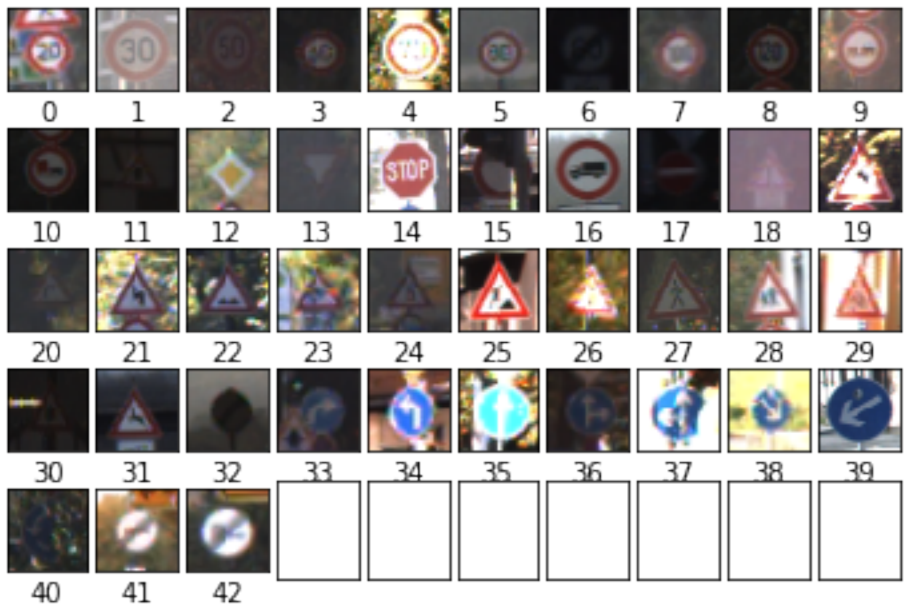
Traffic Sign Classifier

Jingxian Lin

October 6th, 2020

Domain Background

The task of recognizing traffic signs is considered exceptionally challenging. To see why, consider that *even a human* would have trouble distinguishing between images of traffic signs found in life.



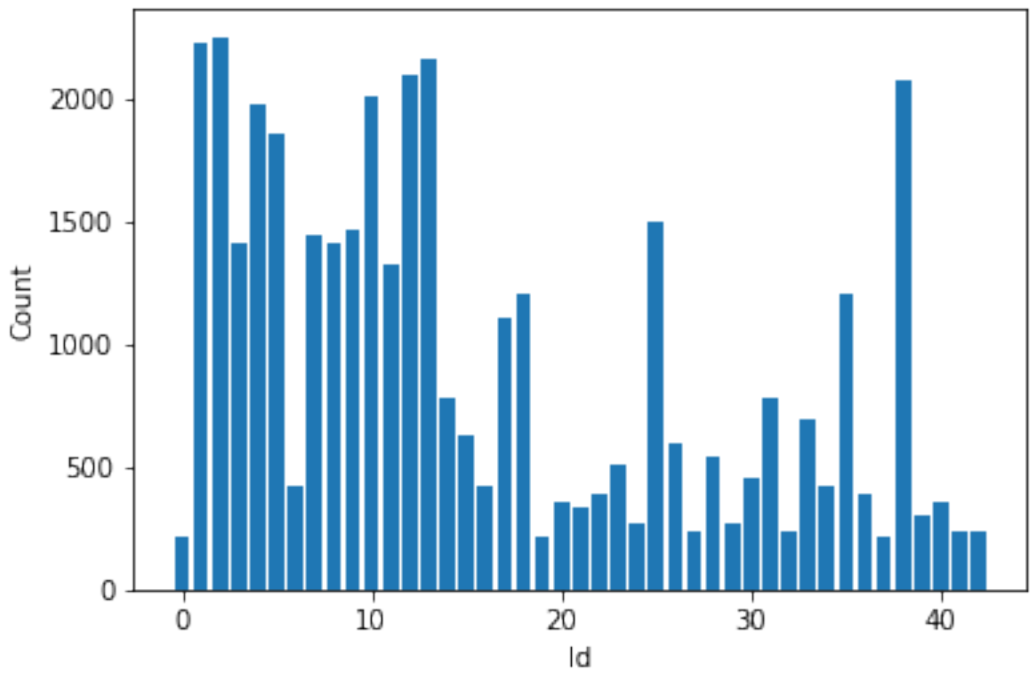
Recalling that convolutional neural networks can be used to classify images with high accuracy and at scale, the goal of this capstone project is to apply deep learning techniques to the classification of traffic signs.

Problem Statement

The main objective of this project will be to build a pipeline to process real-world images. Given an image of a traffic sign, your algorithm will identify an estimate of the traffic sign. This is in fact a classification problem under a supervised learning domain, with the goal to reach a high accuracy.

Datasets and Inputs

German Traffic Sign Dataset is used. There are 39209 training images and 12630 testing images with 43 traffic sign categories. Label distribution (the count of each sign) is plotted below (the shown imbalance between different traffic signs influencing the metric choice and data sampling techniques, a must-have EDA step):



Solution Statement

Use conv_layer, flatten_layer, and fc_layer to create a CNN that can identify traffic sign from images. This way would reach higher accuracy than the following benchmark model, because the architecture is based on the extended model as a feature extractor, and a pooling layer and a fully connected layer are added.

Benchmark Model

Setting aside the fact that the classes are slightly imbalanced, a random guess will provide a correct answer roughly 1 in 43 times, which corresponds to an accuracy of less than 3%. A better benchmark model will be to create a LeNet to classify traffic signs from scratch, which should improve the accuracy.

Evaluation Metrics

The evaluation metric that can be used to quantify the performance of both the benchmark model and the solution model is the accuracy score. Given the number of classes and the distribution of the labels, accuracy is good enough.

Project Design

Data Preprocessing:

Load the data set; Explore, summarize, and visualize the data set; Resize the images to (32, 32, 3) and rescale the images by dividing every pixel in every image by 255; Generate fake data by applying random rotation, random translation, and brightness augmentation; Cache augmented images.

Architecture Constructing:

Split the data into train, test, and validation sets; Define the CNN structure; Specify loss function and optimizer.

Model Training and Evaluation:

Train and validate the proposed model, save the final model parameters, test the model based on the evaluation metric presented above; Use the model to make predictions on new images; Analyze the result of new images, and, finally, provide 3 possible points of improvement.