# Traffic Sign Recognition Using CNN

**Authors**
Hatim Alhazmi Orish Jindal Vaibhav Bishi

## Abstract

One of the critical milestones in the way of our dream of Autonomous Cars is automatic traffic sign recognition. Even before achieving that dream, the traffic sign recognition can assist the driver greatly as sometimes it is difficult for a human to recognize the traffic sign in conditions like rain, fog, lousy windshield visibility, or human factors like drowsiness, near-sightedness, etc.

With this motivation in mind, our project aims to implement real-time traffic sign detection and recognition. The first step is building a machine learning model to classify different traffic signs. A popular way to do it is by using Convolutional neural networks (CNNs), which are widely used for image classification tasks. We implemented a CNN architecture [1] to perform traffic signs recognition.

## 1 Dataset

The dataset used in this project is GTSRB (German Traffic Sign Recognition Benchmark). It is a multi-class, single-image dataset consisting of 43 different classes with 26,727 images in total. The images of each class are taken in different lighting conditions and resolutions. This diversity in the dataset is necessary for training a good model. We divided the dataset into training and validation sets in an 80:20 ratio. The data in the validation set is not used to train the model itself, and thus this provides us with a good measure of the model accuracy. Figure 1 shows a sample of each class.

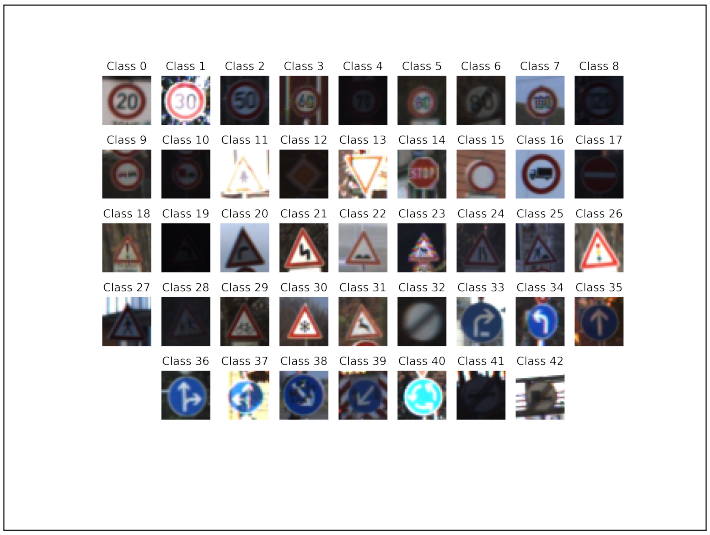



Figure 1: Sample images from each class.

## 2 Methodolgy

Our first task in this project is to build a deep learning model to classify traffic signs. There are various models for performing classification, such as simple machine learning models, SVM, MLP, and CNN. Convolutional Neural Networks (CNN) perform much better in visual data as they can learn more discriminative features in an image. The importance of such a model is to build a reliable traffic sign recognition system to be implemented in an autonomous vehicle or used in driver assistance. This relies on a dashboard camera that gives the real-time video of the environment as an input to the program to recognize traffic signs. This is the main objective of our project. As a precursor to real-time object detection and recognition, we first perform image classification on a set of relevant traffic signs that a vehicle would observe on the road. We implement a custom CNN model to achieve our objective.

CNN consists of multiple layers; there are three main categories: an input layer, an output layer, and a hidden layer that includes multiple convolutional layers, pooling layers, fully connected layers, and normalization layers. The architecture of the CNN model we implemented is shown in Figure 2. It has four convolutional layers, two max-pooling layers, and two fully connected layers.

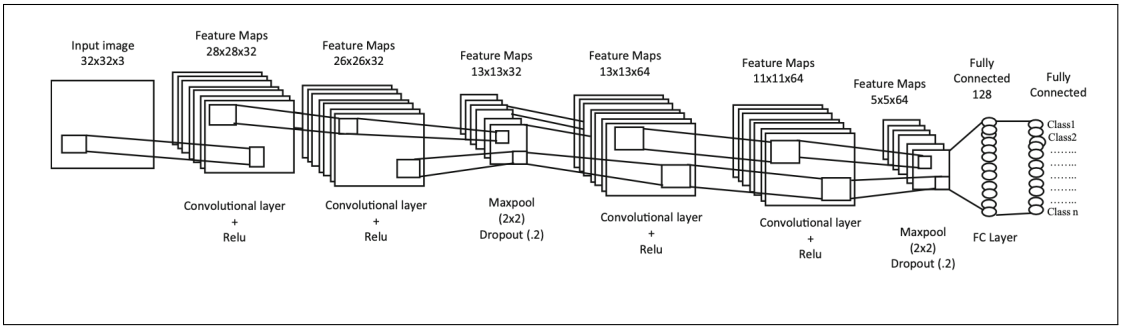



Figure 2: Implemented Model.

## 3 Implementation and Experimentation

There are various ways to implement any machine learning model in Python. One efficient way is to use PyTorch, an open-source machine learning framework based on the Torch library developed by Facebook's AI research lab. We used $nn.Module$ API in PyTorch to implement each aspect of our architecture, such as convolution, max pooling, and fully-connected network layers.

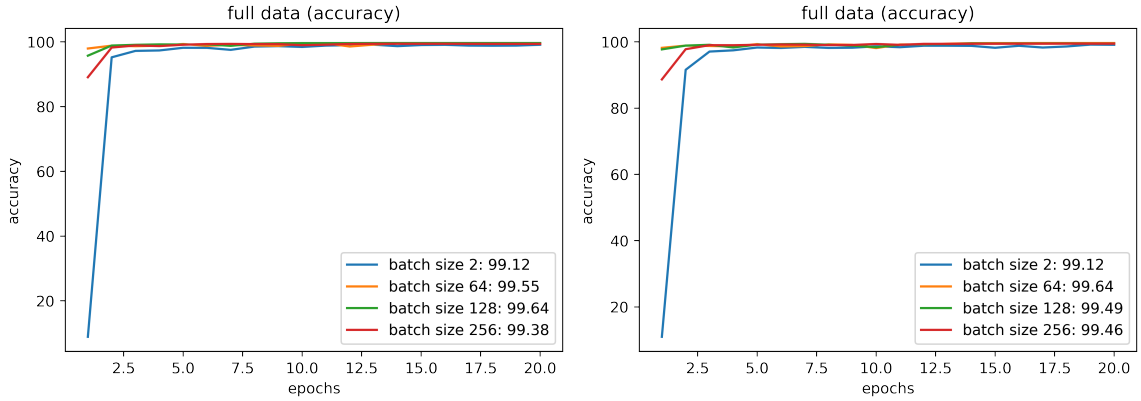
We experimented with different parameters to tune our model for optimal performance in terms of validation accuracy with our dataset. We used four different batch sizes: 2, 64, 128, 256 and four different learning rates: 0.1, 0.01, 0.001, 0.0005. The parameters include batch size, learning rate, optimizer, scheduler, and batch normalization. For the optimizer, we tried Stochastic Gradient Decent (SGD) and ADAM, and we were able to achieve better results with the latter.

We also tried one of the popular models, AlexNet, which resulted in an accuracy of 99.2%. However, the training time was about 10 hours which was much higher than the model that we finally implemented. Here is the link for our code: **click here**.

## 4 Results

We achieved high validation accuracy (above 99%) in our results obtained from different experimentations. One of the reasons is that we have a good quality dataset, i.e., we have a large number of images to train from and a lot of diversity in the images (different resolution, colors, and lighting conditions). Using our model, we achieved an accuracy of 99.64% with the following specifications: batch size=128, ADAM optimizer with a learning rate=0.001, exponential scheduler with decay factor=0.97. The use of batch normalization did not impact the validation accuracy, as shown in Figure 3, but it led to faster training. The effect of the learning rate on the validation accuracy is

shown in Figure 4. Compared to other popular models, such as AlexNet, the architecture of our model is more compact and requires much fewer parameters to learn. Hence, the training and classification time is lower. This is an essential factor for good performance in real-time recognition of traffic signs.




(a) with Batch Normalization.

(b) without Batch Normalization.

Figure 3: Model Validation Accuarcy With different Batch Sizes.

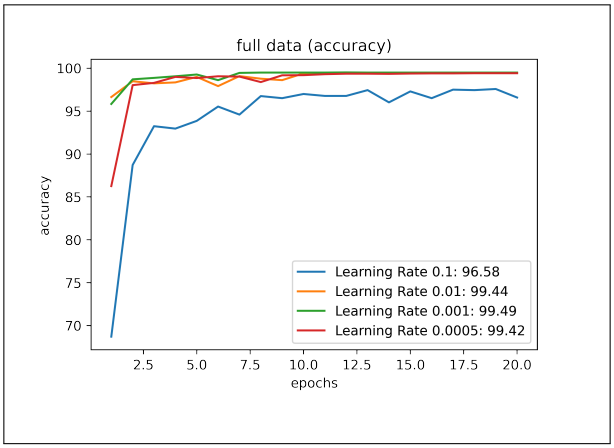



Figure 4: Implemented Model.

## 5 Next target

We will explore popular object-detection models such as YOLO and Faster R-CNN for our final target of real-time detection and recognition of traffic signs. We can further incorporate the classification model that we have implemented into the object detection models. We can also apply data augmentation to increase the strength and diversity of our dataset for better performance.

## References

[1] Jayant Mishra and Sachin Goyal. An effective automatic traffic sign classification and recognition deep convolutional networks. *Multimedia Tools and Applications*, pages 1–20, 2022.