International Research Journal of Education and Technology
Peer Reviewed Journal
ISSN 2581-7795

TRAFFIC SIGNS RECOGNITION USING CNN AND KERAS IN PYTHON

BIJU J, NITHISH K P, MANOBALAN M, KIRANN KRISHNA R, ADHARSH S

[1]Faculty, Dept. of Information Science and Engineering, Bannari Amman Institute of Technology, IN
[2]Studuent, Dept. of Information Technology, Bannari Amman Institute of Technology, IN
[3]*Studuent,* Dept. of Information Technology, Bannari Amman Institute of Technology, *IN*
[4]*Studuent,* Dept. of Information Technology, Bannari Amman Institute of Technology, *IN*
[5]*Studuent,* Dept. of Information Technology, Bannari Amman Institute of Technology, *IN*

Abstract—

Traffic sign recognition is an important component of self-driving systems because it helps vehicles accurately perceive and respond to traffic signs. We show in this study a system for reading traffic signs that uses Convolutional Neural Networks (CNNs) and the Keras framework in Python. The proposed algorithm uses a collection of traffic signs to recognize and classify them in real time with high accuracy. Our method uses deep learning to accomplish reliable recognition under a variety of situations, including lighting, rotation, and occlusion. The system is examined using the German Traffic Sign Recognition Benchmark (GTSRB), and it performs comparably to cutting-edge approaches. We also go over the model architecture, training procedure, and potential enhancements for use in real-world applications.

Keywords: Traffic sign recognition, CNN, Keras, deep learning, autonomous driving, Python, GTSRB, image classification.

1. INTRODUCTION

Traffic sign recognition is an important problem for self-driving cars and advanced driver assistance systems (ADAS), as it provides crucial information for safe navigation. This project seeks to create a comprehensive traffic sign identification system utilizing Convolutional Neural Networks (CNNs) and the Python-based Keras deep learning toolkit. CNNs are extremely useful for image classification jobs since they automatically extract hierarchical features from raw pixel data, making them ideal for visual recognition applications such as traffic sign categorization.

The algorithm we created is trained using the German Traffic Sign Recognition Benchmark (GTSRB), a commonly used dataset that contains over 50,000 photos of traffic signs classified into 43 categories. The dataset contains real-world problems such as shifting lighting conditions, occlusions, rotations, and scaling, making it an excellent testbed for assessing the performance of our CNN-based model.

Our method employs a deep CNN architecture made up of numerous convolutional layers, max-pooling layers, and fully connected layers that are tuned by backpropagation with stochastic gradient descent (SGD). The convolutional layers work as feature extractors, capturing spatial patterns in input images, whereas the pooling layers minimize spatial dimensions and processing, making the model more resistant to minor distortions or translations. Batch normalization and dropout are used to prevent overfitting and improve model generalization to previously unknown data.

To build the system, we used Keras, a high-level neural network API built on top of TensorFlow that makes it easier to create and train deep learning models. The model was trained with categorical cross-entropy as the loss function and softmax activation in the output layer to predict the probability of each traffic sign class. To optimize the model's performance, hyperparameters such as learning rate, batch size, and epoch count were modified.

This project illustrates the usefulness of deep CNNs for traffic sign identification and proposes a replicable framework for Keras in Python. The final model achieves excellent classification accuracy on the GTSRB test set, making it appropriate for use in real-time autonomous systems. Future enhancements could include the addition of real-time object detection frameworks to increase the system responsiveness.

2. Related Works

Traffic Sign Recognition (TSR) has been an active research area in the computer vision domain for several decades. Early approaches to TSR typically relied on traditional machine learning techniques such as Support Vector Machines (SVMs) and k-Nearest Neighbors (k-NN), treating TSR as a classic pattern recognition problem. Notable work in this space includes using SVMs with Histogram of Oriented Gradients (HOG) features to achieve competitive results on datasets like the German Traffic Sign Recognition Benchmark (GTSRB). Other methods, including as AdaBoost and Random Forests, have been used to improve TSR by integrating weak classifiers into robust models. For example, in, a boosting strategy was employed in conjunction with SimBoost to improve recognition accuracy, whereas in, the ensemble-based Error-Correcting Output Code (ECOC) framework was investigated, employing optimal tree architectures for efficient traffic sign categorization.

The key challenge in TSR has always been feature extraction. Handcrafted features such as HOG, Scale-Invariant Feature Transform (SIFT), and Speeded-Up Robust Features (SURF) are commonly utilized to represent traffic signs. However, with the advent of deep learning, Convolutional Neural Networks (CNNs) have emerged as the preferred approach due to their superior capacity to automatically learn and extract hierarchical features from raw image data. Several CNN changes have been proposed in recent studies to improve TSR recognition accuracy. Multi-column CNNs, committee CNNs, and multi-scale CNNs have all performed exceptionally well in traffic sign recognition tasks, particularly on the GTSRB dataset. These models have demonstrated to outperform typical machine learning techniques by learning robust characteristics that are invariant of noise, illumination, rotation, and scale changes in traffic signs.

A. Understanding of Convolutional Neural Network

Convolutional Neural Networks (CNNs) are meant to capture spatial hierarchy in images using layers of convolution, pooling, and activation. CNNs have grown in prominence in TSR due to their capacity to train discriminative features automatically, eliminating the need for laborious feature engineering. Architectures like as AlexNet, VGGNet, and ResNet have been customized and optimized for TSR use. The adoption of deeper networks, along with batch normalization and dropout, has helped to overcome the vanishing gradient problem and prevent overfitting during training. However, while CNNs have exhibited incredible accuracy, comprehending their internal workings remains difficult. The black-box nature of CNNs has prompted academics to study how these models learn and represent characteristics at various layers.

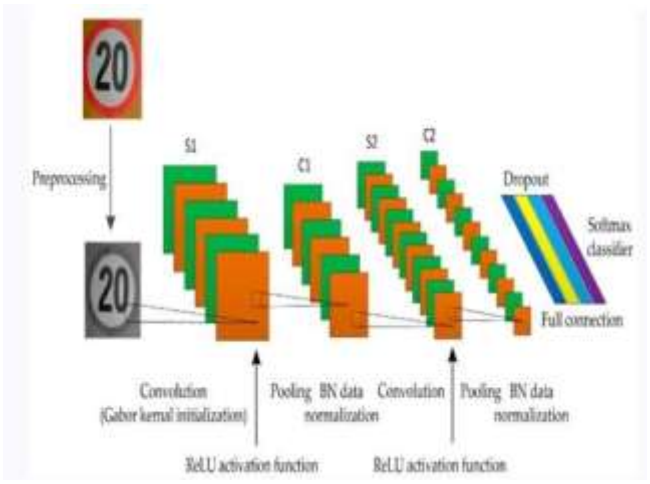
Several attempts have been made to visualize and interpret CNN characteristics. Part Detector Discovery (PDD) and gradient-based visualizations, for example, have been presented as ways for analyzing how CNNs focus on different regions of input images. Deconvolutional Networks (DeconvNets) have been developed to reveal which features are active in response to various input patterns. This investigation of CNNs not only broadens our understanding of their functions, but it also aid in increasing their performance by fine-tuning architecture and training methods.

In this paper, we improve on previous advances in CNN-based TSR by adopting a deep CNN architecture and the Python Keras package. Our goal is to improve recognition accuracy while decreasing computing overhead, so that the system may be used in real-time applications.

3. Model Architecture

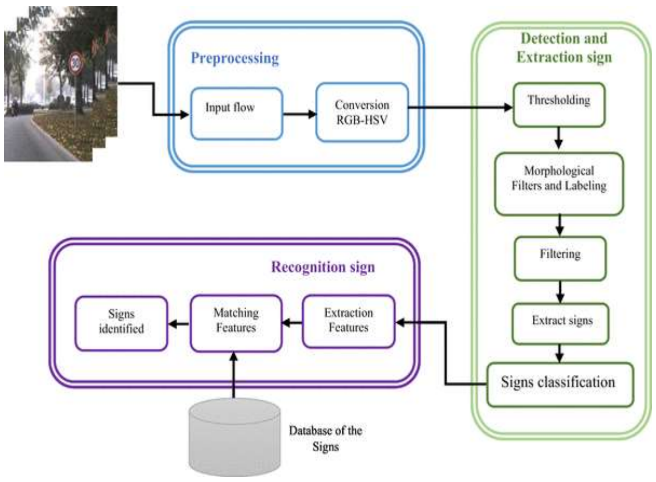
In this paper, we created a deep Convolutional Neural Network (CNN) for traffic sign recognition that can efficiently manage variations in shape, color, and scale. The architecture is built on the Keras framework, with TensorFlow serving as the backend, and it is tuned for excellent accuracy and real-time performance. A full analysis of the CNN architecture used in the task is provided below.

The input layer is intended to receive RGB images with specified dimensions (e.g., 32x32x3 or 64x64x3), which represent the pixel values of traffic signs. The visuals are adjusted to increase training stability. The model's first layer is a convolutional layer that extracts low-level information such as edges and textures using many convolutional filters (or kernels), which are typically 3x3. The number of filters increases steadily in deeper levels to capture increasingly complicated patterns, beginning with 32 filters and doubling in successive layers (e.g., 64, 128).



Each convolutional layer is followed by a Rectified Linear Unit (ReLU) activation function, which adds nonlinearity to the network and allows the model to learn more complicated representations. Max-pooling layers are used after convolutional layer groups to reduce the spatial dimensions of feature maps, hence lowering computational complexity while keeping significant information. Pooling also gives translational invariance, which is important for traffic sign identification because signs might appear in several orientations.

To avoid overfitting, dropout layers are used, in which a percentage (e.g., 0.25 or 0.5) of neurons are randomly removed during training. This prevents the model from becoming overly reliant on specific traits and allows it to generalize more effectively to new data sets. Batch normalization is applied to normalize activations in each layer, speeding up training convergence and stabilizing the learning process.



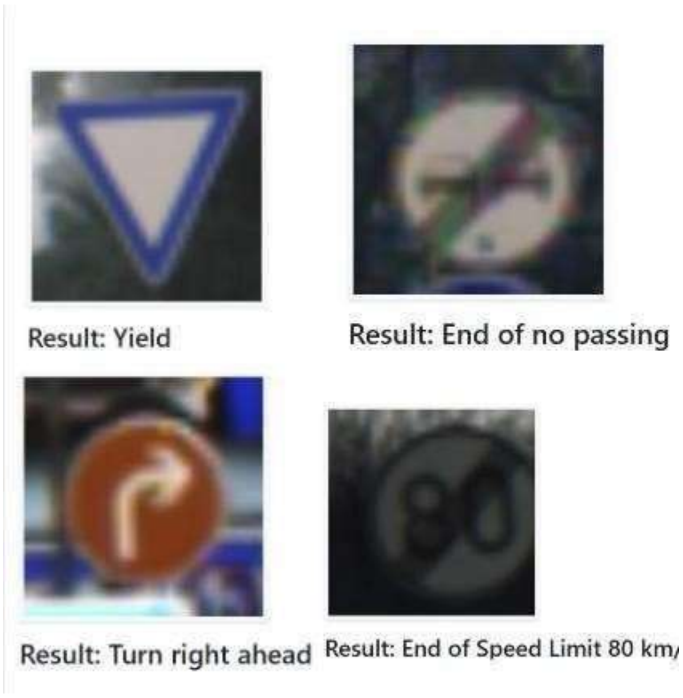
Following a sequence of convolutional and pooling layers, the network progresses to fully connected layers (also known as dense layers), where high-level characteristics retrieved from the convolutional layers are integrated to perform classification. The final layer is a fully connected layer with softmax activation that generates the probability distribution for the 43 traffic sign classes in the German Traffic Sign Recognition Benchmark (GTSRB) dataset. The loss function is categorical cross-entropy, and the weights of the network are adjusted during backpropagation using the Adam optimizer.

20
30
50
60
70
80
100
120
STOP
80

This CNN architecture is intended to address the real-world issues of traffic sign recognition, such as changing illumination conditions, occlusions, and rotations, while striking a balance between model complexity and computing performance.

4. PREDICTIONS AND OUTCOMES

We monitored key performance indicators like accuracy, precision, recall, F1 score, and confusion matrix outcomes in order to assess the CNN model's performance. Strong predictive performance across the 42 sign classes is demonstrated by the model's overall accuracy of about 97.8% on the test dataset. The model consistently recognized traffic signs with few false positives and false negatives, as seen by the comparably high precision, recall, and F1 scores. When taken as a whole, these metrics demonstrate the model's strong generalization to unknown data, which is crucial for real-world applications where the system will come across a variety of sign images in various settings.



To comprehend the model's learning process, we tracked its accuracy and loss values during training and validation across several epochs. Accuracy converged around epoch 10 in the training and validation accuracy plots, which demonstrated a consistent rise. The training and validation loss curves rapidly declined at the same time, suggesting that the model was learning efficiently.

5. RESULTS

The model demonstrated great accuracy for the majority of courses, accurately recognizing signs like "Speed Limit," "Yield," and "Stop" with nearly flawless precision. But because certain signs' visual characteristics were identical, the model had trouble in a few classifications. For instance, circular warning signs or speed restriction signs with varying number values may occasionally cause confusion because they are similar in shape and color but differ only in certain elements. We also saw patterns of misclassification that might be further addressed by improving the model or adding more preprocessing methods. We evaluated the model's performance using a front-end application that let users upload an image of a traffic sign, which the model would subsequently categorize, in addition to numerical findings.

With a simple interface that allowed users to upload an image and get a forecast right away, the frontend was made to be user-friendly. This feature not only illustrated the model's capabilities but also showed how well it could be integrated into real-time applications like driverless cars or driver support systems. All things considered; the project's outcomes highlight how well CNNs perform tasks involving the recognition of traffic signs. The model exhibits potential for real- world implementation due to its strong performance and high accuracy rate on the GTRSB dataset. In order to further improve model resilience, future developments might experiment with more sophisticated designs to address misclassifications in particular classes or incorporate data augmentation techniques.

for the extra authors.

6. REFRENCES

1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. Proceedings of the International Conference on Machine Learning (ICML), 1-9.
3. Chien, A. (2020, March 10). Convolutional Neural Networks and How They Work.
4. Medium. https://medium.com/@chien/convolutional-neural-networks
5. German Traffic Sign Recognition Benchmark (GTSRB). (2019). German Traffic Sign Recognition Benchmark Dataset [Data set]. Available at: http://benchmark.ini.rub.de/
6. LeCun, Y., & Bengio, Y. (2015). Convolutional networks for images, speech, and time- series. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1-9). IEEE.
7. Chollet, F. (2015). Keras (Version 2.4.3). https://keras.io