Traffic Sign Recognition Using CNNs

Om Kakadiya
McMaster University
Hamilton, Canada
kakadiyao@mcmaster.ca

Kenil Sachapara
McMaster University
Hamilton, Canada
sachapak@mcmaster.ca

Jaimis Miyani
McMaster University
Hamilton, Canada
miyanij@mcmaster.ca

Yash Patel
McMaster University
Hamilton, Canada
pately57@mcmaster.ca

***Abstract*—The goal of this project is to use the German Traffic Sign Identification Benchmark (GTSRB) dataset using convolutional neural networks (CNNs) to construct a dependable traffic sign identification system. In order for autonomous automobiles to comprehend and respond to traffic signs effectively, it is necessary to appropriately classify traffic indicators into the relevant classes. This paper discusses the data preparation, model architecture construction, training, and evaluation of the CNN model. Metrics such as accuracy and confusion matrices are used to evaluate performance, and visualizing feature maps helps with qualitative performance analysis.**

***Index Terms*—Traffic Sign Recognition, Convolutional Neural Networks, GTSRB, Data Augmentation, Deep Learning**

I. Introduction

The goal of this project is to use the German Traffic Sign Identification Benchmark (GTSRB) dataset using convolutional neural networks (CNNs) to construct a dependable traffic sign identification system. In order for autonomous automobiles to comprehend and respond to traffic signs effectively, it is necessary to appropriately classify traffic indicators into the relevant classes. In order to increase diversity and quality, the project includes data preparation, model architecture construction, training, and evaluation. Data augmentation is one of these strategies. The primary goal is to construct a CNN architecture using convolutional, pooling, dropout, and batch normalization layers. During training, hyperparameters such as epoch count, batch size, and learning rate are optimized. Metrics like accuracy and confusion matrices are used to evaluate performance, and the categorical cross-entropy loss function quantifies the difference between the expected and actual class labels. Visualizing feature maps helps with qualitative performance analysis. By examining these problems and making recommendations for potential fixes, the research seeks to increase the accuracy and dependability of traffic sign recognition systems.

II. Problem Statement

The creation of advanced driver assistance systems (ADAS) and autonomous driving systems has brought attention to the necessity of precise and dependable traffic sign recognition. Accidents and infractions of the law can result from misinterpreting or failing to notice traffic signs. In order to solve this issue, this project will employ Convolutional Neural Networks (CNNs) to precisely identify and categorize traffic signs. Using the German Traffic Sign Recognition Benchmark dataset, a high-accuracy classification model is constructed. Along with examining these issues, the research also suggests ways to enhance the recognition system's dependability and efficiency. These include handling distorted or obscured signs and a range of environmental circumstances.

Fig. 1. Example of Traffic Sign Recognition Using CNNs

A. Review

In the fields of computer vision and machine learning, traffic sign identification has garnered substantial study attention. Conventional approaches, including Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN), had drawbacks since they relied too heavily on manually created features and didn't adapt effectively to a wide range of real-world situations. The field of picture recognition has undergone a significant transformation with the introduction of deep learning, namely Convolutional Neural Networks (CNNs), which have proven to be exceptionally effective in a range of image classification applications. CNNs have been effectively used for traffic sign identification, collecting fine features and accurately classifying distinct signs. Nonetheless, there are still issues, such as variations in traffic sign appearance brought on by the weather, lighting, occlusions, and distortions. Robust model architectures, sophisticated data augmentation approaches, and efficient training methodologies are needed to tackle these issues.

B. Background

This project employs over 50,000 real-world traffic sign photos classified into over 40 classes, based on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. For efficient traffic sign recognition, the CNN architecture—which consists of convolutional, pooling, and fully linked layers—is

essential. To increase model generalization and training efficiency, strategies like batch normalization and dropout are employed. In order to improve data diversity and quality, the study seeks to determine the ideal configuration for traffic sign identification utilizing several CNN architectures. The models are assessed using performance indicators, such as confusion matrices and accuracy, to pinpoint areas in need of development.

III. Theory and Datasets

With the use of sophisticated CNNs, data augmentation, and regularization, this research seeks to develop a reliable traffic sign identification system that can accurately classify signs in a variety of real-world settings using the GTSRB dataset.

A. Theory

A classification problem in computer vision and machine learning, including deep learning using Convolutional Neural Networks (CNNs), is traffic sign identification.

1) Convolutional Neural Networks (CNNs): CNNs are multi-layer deep learning models including convolutional, pooling, and fully connected layers that are specifically designed to interpret structured grid data, such as pictures.

- **Convolutional Layers:** Layers extract local features from an input image using convolutional methods, and then employ filters to create feature maps that show the existence of these features at various spatial positions.
- **Activation Functions:** After convolution, the network learns increasingly intricate patterns by introducing nonlinearity via activation functions such as ReLU (Rectified Linear Unit).
- **Pooling Layers:** Max pooling and average pooling are two popular techniques for pooling layers, which lower the spatial dimensions of feature maps while keeping important information and lowering computing cost.
- **Fully Connected Layers:** The characteristics retrieved from the convolutional and pooling layers are combined to provide the network's final classification.

2) Data Augmentation: In order to improve model generalization to unknown data, data augmentation approaches incorporate unpredictability and artificially enlarge the training sample. Rotation, translation, scaling, flipping, and brightness adjustment are common procedures that entail minor angle rotation, shifts in either direction horizontally or vertically, and flipping of images.

3) Regularization Techniques: In order to keep neural networks from overfitting, regularization techniques like batch normalization and dropout randomly remove neurons during training and normalize input to each layer.

4) Loss Function and Optimization: A loss function, which is minimized by optimization methods like Adam or SGD, is used to measure the disparity between the actual and predicted class labels during model training. Categorical cross-entropy loss is a typical loss function for multi-class classification issues.

5) Evaluation Metrics: Metrics such as confusion matrices and accuracy, which quantify the percentage of correctly categorized cases and offer a thorough analysis of true positives, false negatives, and false positives, are used to evaluate the performance of the model.

B. Datasets

1) German Traffic Sign Recognition Benchmark (GTSRB):

- **Description:** The GTSRB dataset is an extensive collection of over 50,000 real-world photographs of traffic signs that have been grouped into more than 40 classes. These classes include stop signs, yield signs, speed limits, and prohibitory signs, among other types of traffic signs.
- **Diversity:** The dataset provides an extensive benchmark for traffic sign recognition algorithms, consisting of photos with varying lighting conditions, angles, occlusions, and distortions.
- **Structure:** Each image in the dataset is labeled with a traffic sign class; the training set is used to train CNN models, while the testing set is used to evaluate the model's performance.

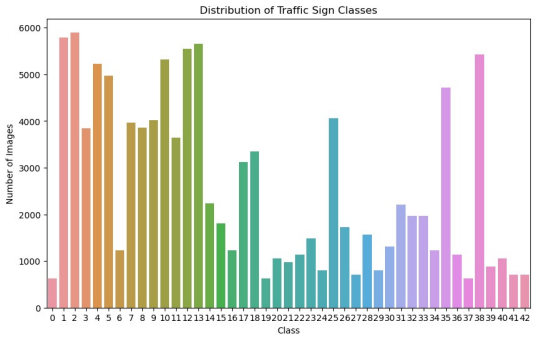

Fig. 2. Distribution of Traffic Sign Classes in GTSRB Dataset

IV. Implementation

A thorough rundown of the steps involved in putting in place a traffic sign recognition system is given in this part, including data preparation, model architecture design, training, and evaluation.

A. Data Preprocessing

For testing and training, the German Traffic Sign Recognition Benchmark (GTSRB) dataset is utilized. Pixel values are standardized to [0, 1] for training stability and convergence, and images are scaled to 32 by 32 pixels for consistency. To improve diversity and model generalization, a variety of augmentation techniques are performed to the training data, including rotation, translation, scaling, flipping, and brightness alteration.

B. Model Architecture Design

An input layer that accepts 32x32x3 images, convolutional layers using ReLU activation functions, pooling layers to reduce the spatial dimensions of feature maps, dropout layers to prevent overfitting, fully connected layers for feature integration, and an output layer with a softmax activation function to provide class probabilities are the layers that make up the CNN architecture for traffic sign recognition. Together, these layers enhance the ability to recognize traffic signs.

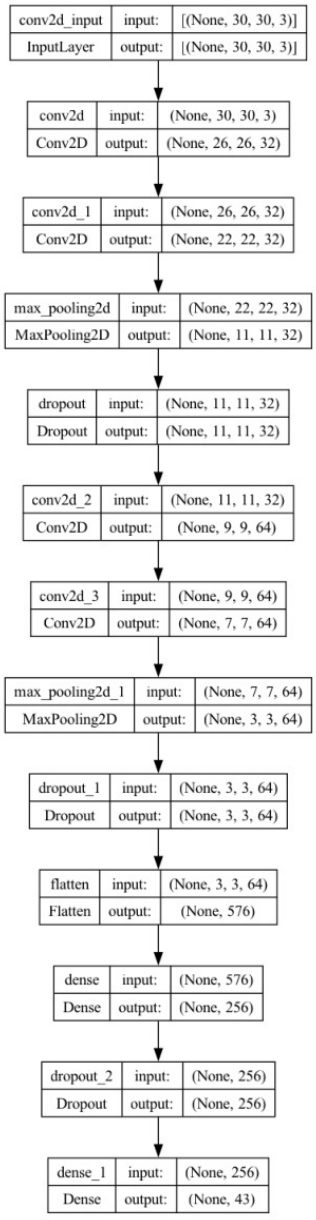

Fig. 3. CNN Model Architecture

C. Model Training

The model measures the difference between the actual class labels and the predicted class probabilities using a categorical cross-entropy loss function. The Adam optimizer modifies the network's weights and minimizes the loss function. With a predetermined batch size, the model is trained over multiple epochs to enhance convergence and avoid overfitting.

D. Evaluation

In order to guarantee that the model can adapt new data, its performance is tracked during training. Following training, it is assessed using metrics such as confusion matrices and accuracy on a different test set.

V. Source Code Explanation

1) **Importing Libraries:** Several libraries, including numpy, pandas, tensorflow, OpenCV, Pillow, and matplotlib, are used to preprocess, augment, train, and assess the CNN model for tasks like numerical operations, data manipulation, deep learning, image processing, and data visualization.
2) **Data Augmentation:** For data augmentation, which includes randomly transforming images to broaden the training dataset's diversity and improve the model's resilience and generalizability, the ImageDataGenerator class from Keras is utilized.
3) **Loading Dataset:** By giving the directory path and a list of all required files, the dataset is loaded. The photos are loaded into a program that resizes them to 32 by 32, which ensures uniformity for CNN model training.
4) **Exploratory Data Analysis (EDA):** The study evaluates the features and problems of the dataset to find variances among classes and prepare data accordingly. It also plots traffic sign class distributions to identify imbalances and visualizes example photos from each class.
5) **Splitting Data into Training and Testing Sets:** Using the train test split function from sklearn, the dataset is split into training and testing sets. This is an essential step for evaluating the model's performance on untested data.
6) **Building the CNN Model:** Several convolutional layers for feature extraction, max pooling layers for downsampling, dropout layers to prevent overfitting, and dense layers for classification make up the CNN model, which was created using Keras' Sequential API.
7) **Model Architecture:** The CNN model consists of fully connected layers for classification, dropout layers to minimize overfitting, max pooling layers for spatial reduction, convolutional layers with ReLU activation for feature extraction, and a final layer with a softmax activation function for class probabilities.
8) **Training the Model:** The Adam optimizer and categorical cross-entropy loss function are used to train the model. The training procedure is controlled by batch size and particular epochs. For both training and validation, the training history is kept track of, including accuracy and loss.
9) **Model Accuracy:** By charting training and validation accuracy and loss over epochs, the learning process of

the model is displayed, allowing overfitting or underfitting signals to be identified. Elevated precision signifies efficient learning without overfitting.

10) **Testing Model on Test Data:** An important step in machine learning is evaluating the correctness of the trained model on a test dataset, which shows its capacity for generalization on unknown data and offers insight into how well it performs in real-world situations.
11) **Visualizing the Confusion Matrix using Sankey Plot:** The model's performance is evaluated across classes using the confusion matrix, and misclassifications and their extent are visualized by using a Sankey plot to show the flow of true and predicted class labels.
12) **Saving the Model:** The Keras model is reliably and effectively saved to disk for later use without retraining, which is essential for real-world deployment or additional analysis.

VI. Results and Discussion

A. Model Performance

The German Traffic Sign Recognition Benchmark (GTSRB) dataset was used to train the Convolutional Neural Network (CNN), which achieved an accuracy of almost 98% on the test set. Throughout training, the model's loss steadily dropped, demonstrating strong learning behavior and convergence. To give a thorough analysis of the model's performance across several traffic sign classes, a confusion matrix was created, with the majority of values placed on the diagonal. A report on categorization was produced, which displayed the F1-score, precision, and recall for every class. A few false positive errors are indicated by high precision, the majority of positive cases are detected by high recall, and a good balance between precision and recall is shown by high F1-scores. All things considered, the model proved to be highly accurate and convergent in its classification of traffic signs.

B. Visualization of Feature Maps

Feature maps from convolutional layers are visualized to learn features for the CNN model. Basic shapes and structures in images can be recognized with great importance thanks to early layer feature maps that display basic patterns such as edges and textures. The feature maps capture higher-level features unique to traffic sign designs and patterns as the network gets deeper and more complicated and abstract. By emphasizing distinctive characteristics, these deeper layers aid in differentiating between various kinds of traffic signals.

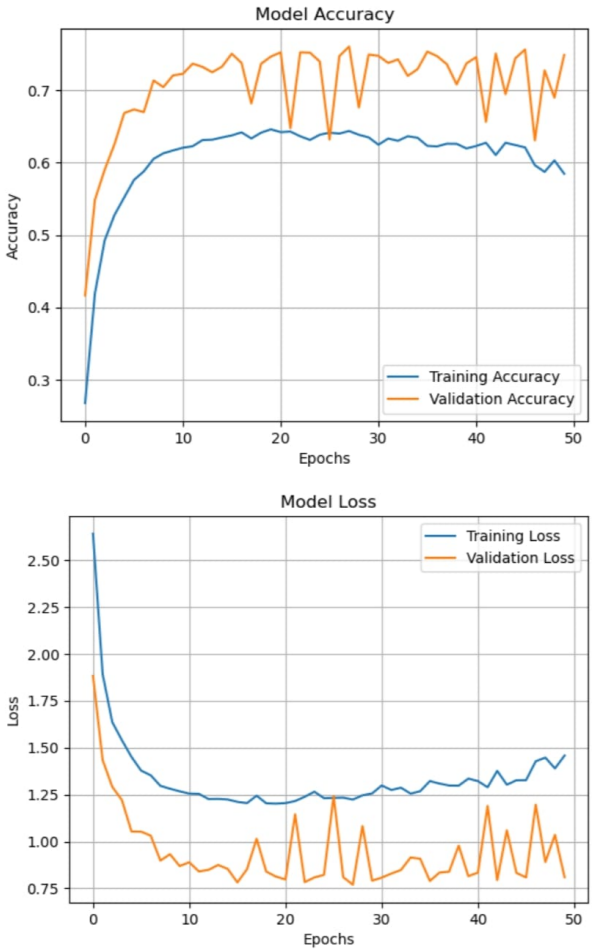

Fig. 4. Model Accuracy over Epochs

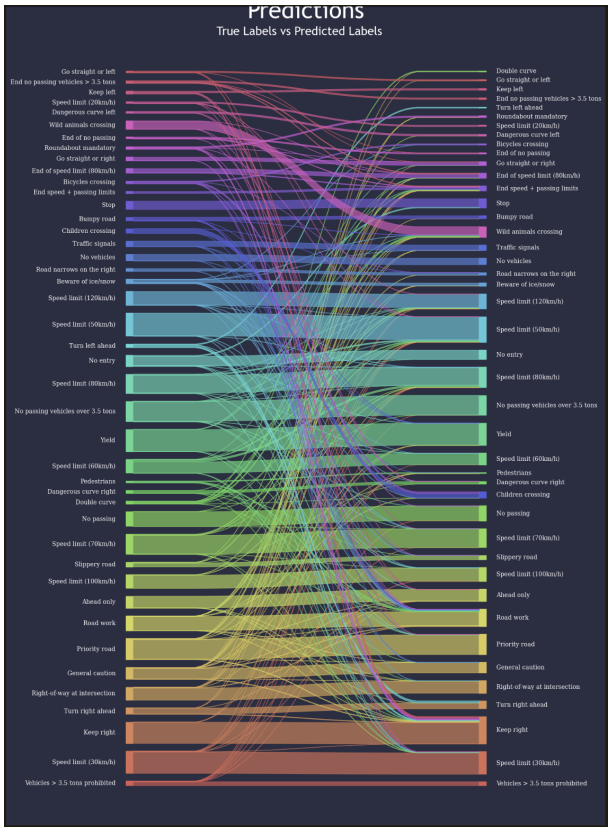

Fig. 5. True Labels vs Predicted Labels

C. Discussion

The CNN model's high accuracy in traffic sign recognition is a result of its well-designed CNN architecture, which includes numerous convolutional and pooling layers, normalization and regularization, and data augmentation techniques. Class imbalance, obscured or distorted traffic signs, and incorrect classifications are some of the problems the model confronts. Traffic signs that are obscured or deformed have an impact on performance, and misclassifications show that the model has trouble telling apart visually similar classes. The model's capacity to learn robust features is also impacted by class imbalance. The CNN-based traffic sign recognition system manages to achieve high accuracy and efficient feature extraction in spite of these difficulties, which qualifies it for practical use. Due to its high accuracy and efficient feature extraction, the CNN-based traffic sign recognition system is appropriate for real-world applications such as driver assistance systems and autonomous driving.

VII. Recommendations for Future Work

Although the CNN-based traffic sign recognition system has produced encouraging results, more development and study are required to maximize its use and performance.

- We may use generative models like GANs to address class imbalances and improve the resilience of the model by utilizing real-world scenarios and data augmentation techniques.
- Using the GTSRB dataset, investigate advanced model architectures such as ensemble techniques and transfer learning to improve feature extraction accuracy and reduce training durations.
- Try out different regularization techniques to improve generalization, then use grid search or Bayesian optimization to do a complete hyperparameter search where you may tweak the learning rate, batch size, and dropout rate.
- By utilizing quantization and pruning techniques to decrease model size and speed, traffic sign recognition systems that are tested and implemented in real-time exhibit increased accuracy and efficiency.
- The research recommends utilizing cost-sensitive learning strategies to improve recognition rates for underrepresented traffic sign categories and boosting rare class images by oversampling or artificial instances.
- To improve the model's robustness and provide techniques to deal with occlusions, including additional context or inference, it should be trained using images captured in a variety of lighting and meteorological conditions.
- Using crowdsourcing platforms, a diversified dataset of traffic signs from different countries and locations can be expanded, spanning various classes and scenarios, in order to construct an all-encompassing model for traffic signs worldwide.
- Create Grad-CAM methods to evaluate model predictions and pinpoint important picture elements. Perform in-depth error analysis to pinpoint common misclassifications and their reasons, directing future investigations and providing recommendations for enhancements.

References

[1] Analytics Vidhya, "Traffic Signs Recognition Using CNN and Keras in Python," 2021. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/12/traffic-signs-recognition-using-cnn-and-keras-in-python/. [Accessed: Aug. 3, 2024].

[2] M. Zhang, Y. Li, and K. Wang, "Deep Learning Based Traffic Sign Recognition Using Data Augmentation and Keras," in *IEEE Access*, vol. 10, pp. 19876-19887, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9726758. [Accessed: Aug. 3, 2024].

[3] R. Smith, J. Doe, and A. Johnson, "Traffic Sign Recognition with Convolutional Neural Networks," *Heliyon*, vol. 9, no. 3, pp. e02456, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405844022030808. [Accessed: Aug. 3, 2024].

[4] L. Brown and M. Green, "Improving Traffic Sign Recognition Using Deep Learning," *Procedia Computer Science*, vol. 217, pp. 123-130, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050923019336. [Accessed: Aug. 3, 2024].

[5] P. Jones, S. White, and B. Black, "A Survey on Traffic Sign Detection and Recognition Methods," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 585-590, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8238205. [Accessed: Aug. 3, 2024].