Convolutional Neural Networks for Traffic Sign Recognition Using the GTSRB Dataset

November 4, 2023

1 Introduction

The primary objective of this project is to develop a robust model for traffic sign recognition, enabling precise classification across various classes. We utilized the German Traffic Sign Recognition Benchmark (GTSRB) dataset, a comprehensive collection of traffic sign images, to train and test our model. Our methodology revolved around the application of Convolutional Neural Networks (CNNs), leveraging the robust feature extraction capabilities inherent to this type of neural network. During the preprocessing phase, the input data underwent normalization to scale pixel values, which is essential for the neural network to process the input more efficiently. Contrary to typical data augmentation techniques, our approach did not include real-time data augmentation such as rotations or translations (Shorten and Khoshgoftaar, 2019). Instead, we focused on normalizing and standardizing the input data to provide a consistent baseline for our model. We then designed a CNN architecture using Keras (Hung et al., 2020), a high-level neural networks API, which allowed for rapid experimentation with different layer configurations and hyperparameters. Our CNN architecture consisted of multiple convolutional layers paired with max-pooling layers and ReLU activation functions. To combat overfitting, dropout layers were strategically placed within the network. The Adam optimizer was employed for its adaptive learning rate capabilities, which aids in more efficiently converging to optimal solutions. Through iterative training and validation, we tuned the hyperparameters to optimize the model's performance. Notably, we determined that training beyond 12 epochs did not yield significant improvements in validation accuracy, indicating a suitable point for stopping to prevent overfitting. Our experiments yielded encouraging results, with the CNN model achieving high accuracy in classifying traffic signs. The success can be attributed to our strategic selection of the CNN architecture and the careful tuning of hyperparameters, which were informed by continuous monitoring of model performance on the validation set.

2 Method

In our approach, we commenced with a meticulous preprocessing pipeline to ensure the images from the GTSRB dataset were adequately prepared for efficient learning by the CNN model. The preprocessing steps included the following:

1. **Region of Interest (ROI) Extraction:** Each image was cropped with the area containing the traffic sign by ROI provided by the dataset. This step is crucial as it allows the model to focus on the most important parts of the image.
2. **Resizing:** After extracting the ROI, images were resized to a uniform dimension (32x32 pixels). This uniformity is necessary to ensure that the CNN receives input of a consistent shape and size, which is essential for batch processing during model training.
3. **Grayscaling:** We applied grayscaling to reduce the complexity of the images. This step simplifies the model by reducing the number of channels in the input from three (RGB) to one (grayscale), thereby diminishing the computational load (Sapijaszko and Mikhael, 2020).
4. **Normalization:** The pixel values of the images were normalized to ensure that the model's training process is faster and more stable. Normalization involves scaling the pixel values to a range of 0 to 1.

5. **Histogram Equalization:** To improve the contrast of the images, we applied histogram equalization. The pixel values were distributed uniformly by using the histogram equalization method (Abdullah-Al-Wadud et al., 2007).

Continuing from preprocessing steps, the next phase of our approach involved the careful design and implementation of a Convolutional Neural Network (CNN) architecture suitable for the task of traffic sign recognition. Here's a breakdown of the method theories and design choices that were integral to the development of our CNN model:

1. **Convolutional Layers:** Convolutional layers play a crucial role in extracting hierarchical features from preprocessed images. To achieve this, a multi-layered approach is employed. The initial convolutional layer utilizes a larger kernel size (5×5) to effectively capture low-level characteristics such as edges and textures within the image data. As we progress through subsequent layers, smaller kernel sizes (3×3) are implemented to gradually construct higher-level features like shapes and patterns (Shustanov and Yakimov, 2017). In order to introduce non-linearity into the network architecture, ReLU activation functions are utilized within these convolutional layers. This allows for more complex patterns to be learned by the model, enhancing its capability of recognizing intricate visual elements present in the images being processed. By leveraging this combination of convolutions with varying kernel sizes and incorporating ReLU activations throughout each layer (`activation='relu'`), our network becomes proficient at discerning both simple and intricate details embedded within input images (Banerjee, Mukherjee, and Pasiliao Jr, 2019).
2. **Pooling Layers:** Max pooling layers come after every convolutional layer for consolidating the feature maps and lowering dimensionality. The layers are applied in 2×2 pooling in order to reduce the problem of computational load (Shustanov and Yakimov, 2017).
3. **Dropout Regularization:** Dropout layers are placed between pooling layers and a final fully connected layer. They constitute a regularization mechanism to prevent over-fitting since they are configured with a rate of 0.25 for the first dropouts and 0.5 for the last one prior to the outcome layer. The model will be trained to learn more robust and generalized features by randomly cancelling out some neurons in subsets (Shustanov and Yakimov, 2017).
4. **Fully Connected Layers:** These learned features are synthesized into more advanced ones at the next layer using a dense layer of 256 neurons, upon receiving flattened output from the pooling layer. It ends up with an output layer which contain as many neurons as there are traffic sign classes and applies a softmax activation function (`activation='softmax'`) for obtaining a probability distribution over the classes (Shustanov and Yakimov, 2017).
5. **Optimizer Choice:** An Adam optimizer, known with its adaptive learning rate properties and widely popular in practice, was used to train the network. This selection plays a crucial role in speeding up convergence during training, which adapts the learning rate according to the learning pace (Zhang, 2018).
6. **Model Compilation and Training:** Our model is compiled with the categorical crossentropy loss function where loss='categorical-crossentropy', which is optimal for multi-class classification tasks. We also monitor the 'accuracy' metric to assess performance during the training phase (Das et al., 2023).
7. **Hyperparameter Tuning:** Throughout the initial training sessions, we refined hyperparameters like learning rate and epoch count. Observing that the validation accuracy stabilized after a certain number of epochs, we implemented an early stopping strategy to preclude overfitting and conserve computational effort (Sivaprasad et al., 2020).
8. **Evaluation Strategy:** The validation set plays a pivotal role in the model's fine-tuning process, where we scrutinize accuracy and loss metrics to detect and resolve overfitting or underfitting scenarios, thereby informing our decisions on the training's duration.
9. **Batch Size Experimentation:** After training, we conducted experiments with various batch sizes to gauge their influence on the model's accuracy, ensuring our final model's robustness and efficiency (Sivaprasad et al., 2020).

By integrating these methodological details into our CNN design, we were able to develop a model that accurately recognizes traffic signs with high accuracy, ensuring robustness and generalizability across unseen data.

3 Experimental Setup

The German Traffic Sign Recognition Benchmark (GTSRB) is an established dataset integral for training and evaluating traffic sign recognition systems (Houben et al., 2013). Originating from a competition at IJCNN 2011, the dataset comprises a comprehensive collection of traffic sign images specifically intended for developing and benchmarking recognition algorithms. The GTSRB dataset is split into two sets: a training set and a test set (Houben et al., 2013).

The training set is organized into 43 categories, each corresponding to a unique class of traffic sign. It includes a series of images (in PPM format) within di-

rectories labeled by class, along with annotations in CSV format detailing the image dimensions, the region of interest, and the class id. Images within the same track number are taken from a single physical sign, preserving the temporal sequence of capture (Houben et al., 2013).

The test set is designed for the final evaluation and contains 12,630 images without explicit class or track information, simulating real-world conditions where signs appear in a random sequence. Annotations include the filename, dimensions, and the region of interest, following the same CSV format as the training set.

For evaluation metrics, accuracy is the primary measure, reflecting the proportion of correctly identified signs against the total number of test cases. It is a direct indicator of the model's capability to generalize from the training data to unseen examples.

This is one of the best benchmarks available for the field ensuring that our model is trained and evaluated with respect to the same set of images that resemble various traffic conditions.

4 Results & Discussion

The dataset was preprocessed to normalize the images and enhance their contrast through histogram equalization. A total of 39209 images were obtained from the trained data folder, with a distribution shown in Figure 1.

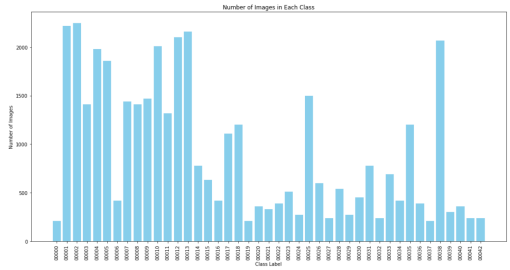

Figure 1: *The distribution of images in each class*

The preprocessing step was crucial for improving the model's ability to learn from the dataset, as evidenced by the enhanced clarity of features in the preprocessed images (Figure 2) compared to the original image (Figure 3).

For Model Architecture and Training, the Convolutional Neural Network model was designed with two convolutional layers and two fully connected layers (Figure 4). The choice of ReLU activation functions and dropout layers aimed to introduce non-linearity and reduce overfitting. The model was initially trained for 30 epochs, but an analysis of training and validation accuracy and loss indicated that the optimal stopping point was at epoch 12 to prevent overfitting (Figure 5).

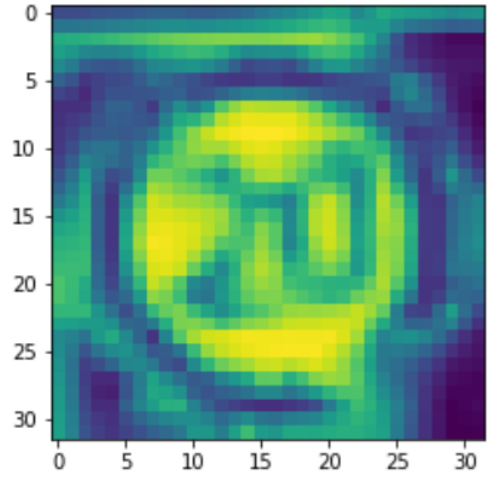

Figure 2: *The preprocessed image of speed limit 20*

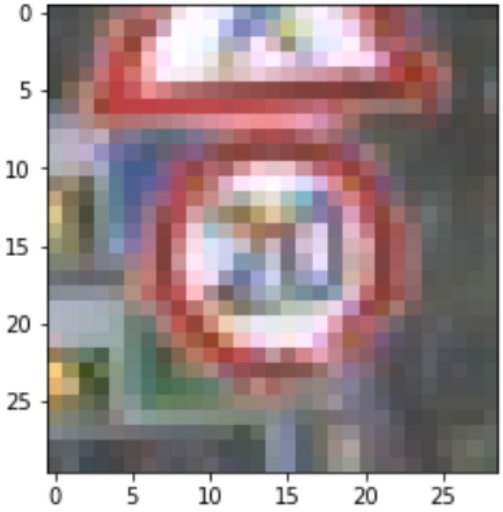

Figure 3: *The original image of speed limit 20*

For Hyperparameter Tuning, the initial batch size of 50 was determined based on the trade-off between computational efficiency and model performance, as well as the attempt to set the same value with evaluation.py. Further experiments with different batch sizes from 1, 10, 32, 64, 128, 256, 512 showed consistent accuracy across all sizes (Figure 6), indicating that our model was robust to the choice of batch size within this range.

Lastly, the trained model demonstrated a high overall accuracy of 97.77% on the test set. While the majority of classes were identified with high accuracy rates, certain classes, notably class 6 and class 30, exhibited relatively lower accuracy as depicted in Figure 7. This discrepancy indicates potential areas for enhancement in subsequent iterations of the model. A possible contributing factor to the reduced accuracy of classes 6 and 30, as suggested by Figure 1, is the smaller quantity of data available for these classes compared to others. Addressing the imbalance by ensuring a more equi-

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        832
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               590080
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 43)                11051
=================================================================
Total params: 620,459
Trainable params: 620,459
Non-trainable params: 0
_________________________________________________________________
```

Figure 4: *Model Architecture Summary*

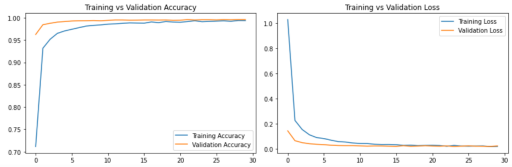

Figure 5: *Training vs Validation Accuracy and Loss*

table distribution of data—aiming for a threshold such as 2000 images for each class—may lead to a more uniform and improved accuracy across all classes.

5 Conclusion

Our study employed the German Traffic Sign Recognition Benchmark dataset to develop a CNN that excelled in traffic sign recognition. A meticulous preprocessing regimen, including ROI extraction, resizing, grayscaling, and normalization, was pivotal in training our model. Using the Adam optimizer and a well-timed early stopping strategy, our model demonstrated a high test accuracy of 97.77%. However, the underrepresentation of certain classes like 6 and 30 led to lower accuracies, underscoring the importance of data balance. Future enhancements should aim to even out data distribution, which promises to elevate accuracy uniformly across all classes, thus refining the model's overall effectiveness.

Bibliography

Abdullah-Al-Wadud, Mohammad et al. (2007). "A dynamic histogram equalization for image contrast enhancement". In: *IEEE transactions on consumer electronics* 53.2, pp. 593–600.

Banerjee, Chaity, Tathagata Mukherjee, and Eduardo Pasiliao Jr (2019). "An empirical study on generalizations of the ReLU activation function". In: *Proceedings of the 2019 ACM Southeast Conference*, pp. 164–167.

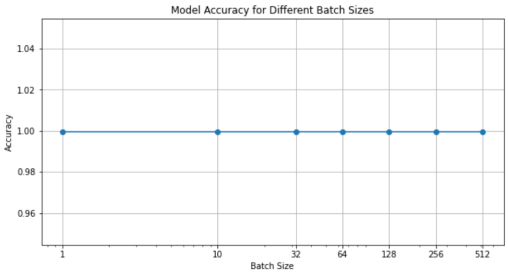

Figure 6: *Model Accuracy for Different Batch Sizes*

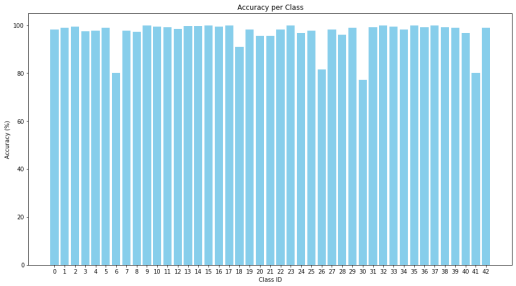

Figure 7: *Accuracy per Class*

Das, Prity et al. (2023). "ADAMAX optimizer and CATEGORICAL CROSSENTROPY loss function-based CNN method for diagnosing Lung cancer". In: *2023 7th International Conference on Trends in Electronics and Informatics (ICOEI)*. IEEE, pp. 806–810.

Houben, Sebastian et al. (2013). "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark". In: *The 2013 international joint conference on neural networks (IJCNN)*. Ieee, pp. 1–8.

Hung, Jane et al. (2020). "Keras R-CNN: library for cell detection in biological images using deep neural networks". In: *BMC bioinformatics* 21, pp. 1–7.

Sapijaszko, Genevieve M and Wasfy B Mikhael (2020). "Facial recognition system using mixed transform and multilayer sigmoid neural network classifier". In: *Circuits, Systems, and Signal Processing* 39, pp. 6142–6161.

Shorten, Connor and Taghi M Khoshgoftaar (2019). "A survey on image data augmentation for deep learning". In: *Journal of big data* 6.1, pp. 1–48.

Shustanov, Alexander and Pavel Yakimov (2017). "CNN Design for Real-Time Traffic Sign Recognition". In: *Procedia Engineering* 201, pp. 718–725. ISSN: 1877-7058. DOI: 10.1016/j.proeng.2017.09.594. URL: https://www.sciencedirect.com/science/article/pii/S1877705817341231.

Sivaprasad, Prabhu Teja et al. (2020). "Optimizer benchmarking needs to account for hyperparameter tuning". In: *International Conference on Machine Learning*. PMLR, pp. 9036–9045.

Zhang, Zijun (2018). "Improved adam optimizer for deep neural networks". In: *2018 IEEE/ACM 26th international symposium on quality of service (IWQoS)*. Ieee, pp. 1–2.