COMP47650–Deep Learning Project

Zairui Zhang 19209905

May 3, 2024

Abstract

This study employs convolutional neural networks (CNNs) for traffic sign recognition using a dataset from Kaggle[1]. We focus on optimizing image preprocessing and model parameters to enhance the accuracy of traffic sign classification. Through systematic adjustment of hyperparameters and model architecture, The primary objective is to develop a model capable of identifying various traffic sign types effectively even if the image quality is bad.

1 Introduction

Image classification is a crucial technique in the analysis of remote sensing images, serving a wide array of applications including environmental monitoring, agricultural management, land use and urban planning, surveillance, geographic mapping, disaster response, and object detection. This field has garnered significant interest within the remote sensing community, positioning it as a focal point of contemporary research[2].

Traffic signs are the input data used in this project to train the model. The precise recognition and categorization of traffic signs are essential to ensuring road safety for the development of autonomous driving[3]. The dataset used in this study is taken from Kaggle and is quite diverse in its different images of traffic signs. To improve the model training outcome, the project thoroughly preprocessed these images for standardized input data, like rescaling, brightness changing, normalization, and rotation techniques.

The main part of our approach is experimental tuning in terms of CNN architectures: we vary a set of hyper-parameters and apply different activation functions and regularization techniques. All these steps aim to optimize the neural network so that the overfitting is reduced and the generalization on unseen data improves.

Our main aim is not only to achieve high accuracy in the classification of traffic signs but also to contribute towards advanced image recognition by proposing effective techniques to tackle the challenges of deep learning. The hope is that the result will be a set of best practices and model configurations that shed light on how to apply CNNs to real-world image-recognition tasks with appropriate efficiency.

2 Related Work

In the dataset, the author has carefully preprocessed and classified the data into nine distinct pickle files. These files are systematically organized based on color encoding—RGB and grayscale—and incorporate various normalization strategies to optimize model training. This comprehensive preparation ensures that the datasets are immediately ready for training purposes.

2.1 Preprocessing potential strategy

Additionally, the dataset includes the original, unprocessed data, providing flexibility for custom preprocessing. This allows us to explore various preprocessing techniques that might be suited to specific project needs. The available techniques for potential implementation include:

1. Color Transformations: Color Transforms from RGB to Grayscale. Reducing the complexity of computation from required visual features for model classification.
2. Normalization: Making uniform the "values of the pixels" across the dataset to improve the performance and stability of a neural network during training.
3. Equalization: Equalizing dataset scale for different classes helps model training efficiency and reduces complexity.
4. Data augmentation: techniques such as rotation, scaling, and change in brightness are used to impersonate a wider reality and make the model more robust to overfitting.
5. Shuffling in random order to avoid sequence learning.

2.2 Convolutional neural networks

Convolutional neural networks(CNN) are popular and modern uses of pattern recognition in computer vision. For each picture, a CNN processes it through a series of convolutional layers. It has some number of trainable filters at each of the layers. The filters slide on the image, and every movement of the filter in the image is an element-wise multiplication with the part of the image of this filter. The products sum to give a feature map where the presence of spatial locations of certain features within the image. The layers of convolutions are of key importance in image analysis since these can identify all such features that are located at different places in the image. One may mathematically describe the operation of convolution that forms the central core of the process[4].

Given an image I and a filter K, the convolution operation F at a position (x, y) is defined by the equation:

$$F(x,y) = (I * K)(x,y) = \sum_{i=-a}^{a} \sum_{j=-b}^{b} I(x-i, y-j) \cdot K(i,j)$$

where $I(x, y)$ represents the pixel values of the image, $K(i, j)$ denotes the coefficients of the filter, and the summations extend over the dimensions of the filter.

This operation mathematically enhances patterns and features in the input image efficiently, making tasks like edge detection or texture recognition easier, which are necessary for further analysis, such as object detection or classification. This means that CNNs can extract features to a very large extent, automatically, with very little or even no requirement for manual feature engineering; this then implies power in a wide range of automated visual tasks.

3 Experimental setup

3.1 Data Preprocessing

We first download data from Kaggle and load the original images into train, valid, test sets. The whole dataset has 51839 images which split 67% for training, 9% for valid, and 24% for test. There are 43 classes of traffic signs. The distribution plot(Figure1) shows different types of signs that the traffic is imbalanced and poses heavy challenges to do the classification task, our first task was to balance all classes in the same sample size.

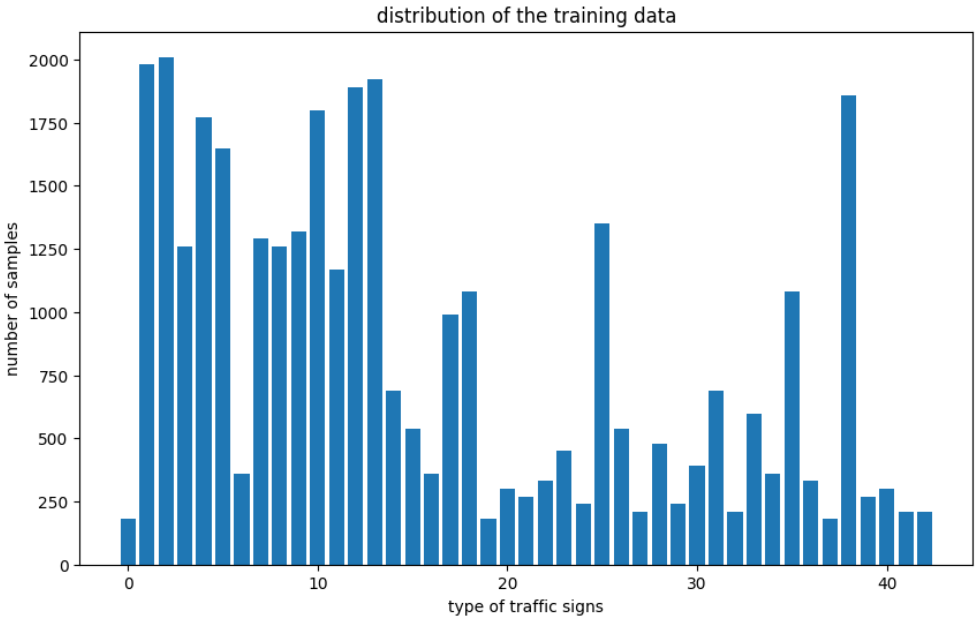

Figure 1: Class distribution

After equalization, the training set has increased to 86989 samples, which

means there are 2023 examples for 43 types of traffic signs. When observing some random samples of traffic signs we found that some images were dark or overexposed and some had the wrong shooting angle(Figure2).

Figure 2: random exampls

To address the variability in lighting conditions and orientation of traffic sign images in our dataset, we use three specific preprocesses: brightness adjustment, rotation correction, and contrast enhancement. These functions help normalize the dataset, thereby improving the robustness of our training model.

For brightness enhancement, the RGB image has been first converted into Hue Saturation Value (HSV) color space, which is more effective for brightness manipulation. A further step is to multiply the brightness (Value channel) by a random factor between the bounds specified for normalization so that the pixel values remain valid. At last, convert the image back to RGB.

Then, we perform a random rotation around its center in the range using the affine transformation capabilities of OpenCV. This correction will process the alignment of the traffic signs uniformly towards the classifier training data and hence will improve accuracy.

Under different lighting conditions, the features of the image will be poorly visible and distinguishable. In the process, we use a method called Contrast Limited Adaptive Histogram Equalization (CLAHE)[5], which is greatly powerful in processing the contrast of traffic sign images without making too much alteration to the global brightness.

After the preprocessing steps, we implement one-hot encoding for the traffic sign labels to facilitate the classification task. This process converts categorical integer labels into a binary matrix representation, this is more suitable for neural network training. The images of original and preprocessed are also shuffled and integrated to form a training set. The shuffle process provides diversity in training examples, which is most important for building the robustness of the model and generalizing power to get good classification accuracy.

Figure 3 illustrates the efficacy of contrast enhancement: the preprocessed image on the right exhibits sharper details and more colors compared to the original on the left, improving feature distinguishability for further analysis.

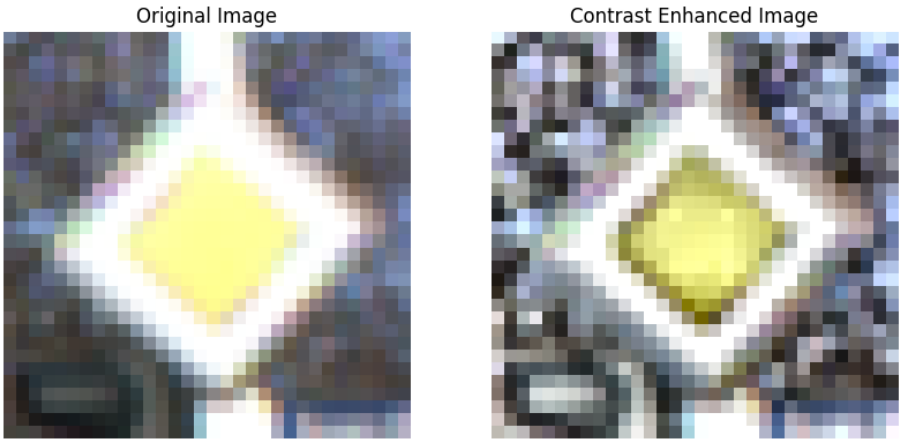

Figure 3: Comparison after preprocessing

3.2 CNN Model

3.2.1 Network Architecture and Hyperparameter

Compilation and Optimization

Optimizer: We use the Adam optimizer which contains the learning rate adaptation method, which is very efficient, especially in datasets with sparse gradients and a diversity of image features.

Loss Function: categorical_crossentropy is a great loss function for the categorical classification tasks and further compels the predictions from the model to the one-hot encoded target vectors.

Metrics: The performance of the model will be majorly based on accuracy. From the training and validation process, the measure of the success of the classification can be obtained directly.

During the model training phase, we use an early stopping[6] callback and set a maximum of 100 epochs to minimize computational waste. This halts training when performance ceases to improve, effectively preventing unnecessary processing. We utilize a batch size of 256 to enhance training speed, optimizing computational efficiency. Additionally, the training process is accelerated by GPU capabilities enabled by CUDA[7], significantly speeding up computations and the overall learning process.

The table3.2.1 below summarizes the convolutional neural network architecture used for classifying traffic signs.

Layer (Type)	Configuration	Output Shape	Parameter
Conv2D (conv2d)	32 filters, 5x5, ReLU	(None, 28, 28, 32)	2,432
MaxPooling2D (max_pooling2d)	2x2	(None, 14, 14, 32)	0
Conv2D (conv2d_1)	64 filters, 3x3, ReLU	(None, 12, 12, 64)	18,496
MaxPooling2D (max_pooling2d_1)	2x2	(None, 6, 6, 64)	0
Dropout (dropout)	rate=0.5	(None, 6, 6, 64)	0
Flatten (flatten)	–	(None, 2304)	0
Dense (dense)	150 neurons, ReLU	(None, 150)	345,750
Dropout (dropout_1)	rate=0.5	(None, 150)	0
Dense (dense_1)	43 neurons, Softmax	(None, 43)	6,493

Table 1: Detailed architecture of the CNN model used for traffic sign classification.

These Hyperparameters include the sizes of filters in convolutional layers, dropout rates, dense layer configuration, etc. And they play a key role in deciding how the model performs and how efficiently it does. It represents a model optimized for powerful performance in classifying different traffic sign images through systematic adjustments and empirical validation, which demonstrates significant potential for real-world applications in automated image recognition systems.

4 Results

Following training, we assessed the model using a test set that achieved an accuracy of 0.961 and a loss of 0.156. The graph illustrates the training and validation loss and accuracy of a CNN model over 32 epochs.

In the beginning, the loss curve for both training and validation falls rapidly within the first few epochs. The rapid decrease shows that the model is learning fine from the training data by making proper adjustments in the weights to minimize the error rate. In the stabilization phase, it shows post-initial descent, and the loss curves in both training and validation loss approaches attain a stable state. The leveling off suggests that it has reached a point where further training doesn't decrease the loss, meaning it's already in an optimal learning state.

For the accuracy plot, it improves immediately Corresponding to the initial drop in loss, there is a sharp increase in both training and validation accuracy. This rapid improvement indicates that the model is compatible with correctly classifying the training data. Training and validation accuracy increase initially

with epochs and then both plateau to a good level.

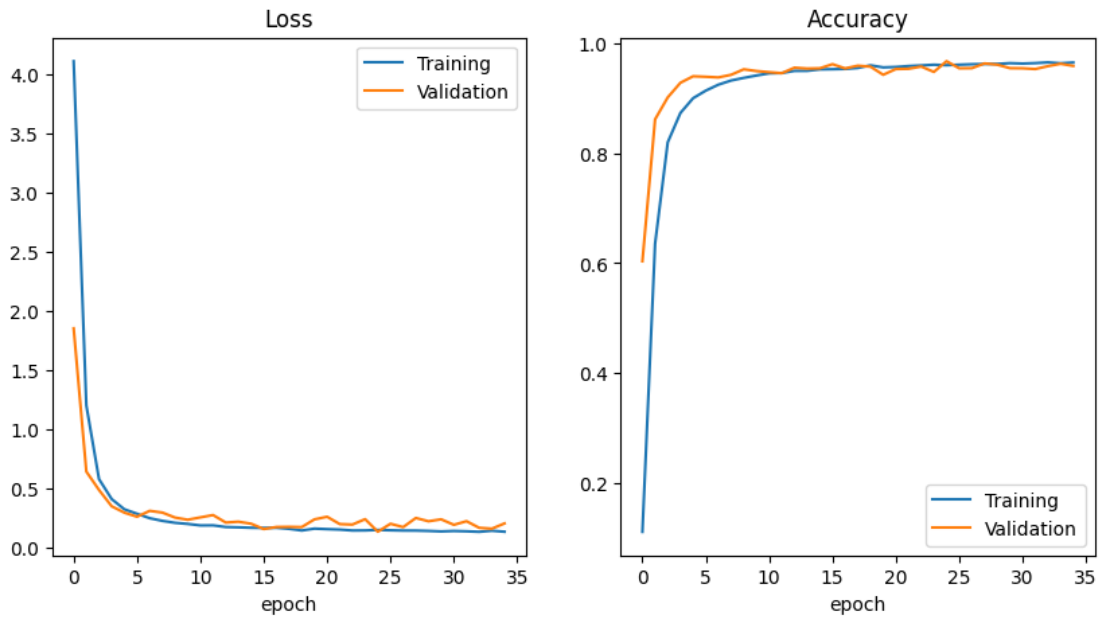

Figure 4: Loss and Accuracy

The architecture of LeNet-5 was also used at the beginning. It was already very high in accuracy during training: 0.994, with 16 epochs, but on the validation set, it was 0.916. When tested on the test set, the accuracy was 0.91, which indicated potential overfitting. These performance results show that if overfitting or poor generalization to unseen data happens, the validation accuracy would jump much during training. Therefore, not deployed in practice. A more fine-tuned parameter model, or in the case of a deeper search for better architecture, has to be pursued to get performance that secures stability in accuracy across different data sets.

5 Conclusion

In summary, This project was first focused on improving the image preprocessing techniques and optimizing the hyperparameters of the CNN models for better performance. It had a function that changed images into grayscale form at the pre-processing stage from the author. Nevertheless, color information in traffic signs, such as white, red, and blue, is very important for finding the right classification of traffic signs. Rather, operations like data equalization technique, brightness correction, and rotational corrections were performed in the best form

to make the images ready for training. Regarding the CNN architecture, the version of the traditional LeNet-5 structure was tested, but it was not satisfactory for recognition. Some changes in the filter sizes and activation functions were taken into account for performance improvement, where the effect of ReLU was stable and more effective in this application. Besides, many of the optimizations were done, all toward achieving more efficiency and smoothness in the training process.

In this context, more advanced architectures, such as GoogleNet's InceptionV3, could be considered for further increase in accuracy and robustness. Further work can attempt different activation functions, like sigmoid or ELU, trying to establish which among them would be best for image classification tasks with the approach tailored to cover the specific difficulties and features of recognition of traffic signs. This forward-looking aim seeks to make use of cutting-edge techniques in an unending attempt to improve the performance of the models given complex classification scenarios.

References

1. *Traffic Signs Preprocessed* `https://www.kaggle.com/datasets/valentynsichkar/traffic-signs-preprocessed?select=label_names.csv`.
2. Deepan, P. & Sudha, L. Object Detection in Remote Sensing Aerial Images: A Review. *International Journal of Scientific Research in Computer Science Applications and Management Studies,* 1–8 (2018).
3. Hindarto, D. Enhancing Road Safety with Convolutional Neural Network Traffic Sign Classification. *sinkron* **8,** 2810–2818 (Nov. 2023).
4. Sichkar, V. & Kolyubin, S. Effect of various dimension convolutional layer filters on traffic sign classification accuracy. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics* **19,** 546–552 (June 2019).
5. *Image Contrast Enhancement Using CLAHE* `https://www.analyticsvidhya.com/blog/2022/08/image-contrast-enhancement-using-clahe/`.
6. *EarlyStopping* `https://keras.io/api/callbacks/early_stopping/`.
7. *CUDA* `https://developer.nvidia.com/cuda-toolkit`.