Resnet for traffic sign classification

Alejandro Salamanca Ruiz
asalama4@jhu.edu

Abstract

In this paper, we propose using a ResNet-based convolutional neural network (CNN) for traffic sign classification on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. We address the challenges of class imbalance in the dataset by applying data augmentation techniques, including random transformations such as color jittering and Gaussian blurring. The performance of the proposed model is evaluated on both the original and augmented datasets. Our results indicate that the augmented dataset yields significant improvements in the model's performance, achieving a test accuracy of 95.62

1 Introduction

In the last decade, deep learning techniques have shown great promise in various computer vision tasks, including image classification. As shown in [5], deep convolutional nets have brought many breakthroughs in processing images. Residual Networks [2] have successfully achieved state-of-the-art results in many image classification challenges. This paper proposes a ResNet-based CNN for traffic sign classification, explicitly focusing on the German Traffic Sign Recognition Benchmark (GTSRB) dataset. This dataset comprises a diverse set of traffic sign images with varying sizes, orientations, and lighting conditions, making it a challenging benchmark for evaluating the performance of the proposed model.

One of the main challenges in training a model on the GTSRB dataset is the class imbalance, where certain classes have significantly fewer samples than others. To address this issue, data augmentation techniques are employed to balance the dataset and enhance the model's ability to generalize to new samples. Transformations, such as color jittering and Gaussian blurring are used to finally assess the model's performance in a test set.

The main contributions of this paper are as follows:

1. An original ResNet-based CNN architecture for traffic sign classification, demonstrating its effectiveness on the GTSRB dataset.
2. Experiments to investigate and assess the impact of class imbalance in the dataset. The paper presents a data augmentation procedure to mitigate its effects on the model's performance.

Two experiments were performed using the same network architecture. In the first one, the network is trained 40 epochs without any data augmentation process, and its performance is tested over a test data set. The second one uses data augmentation techniques to upsample classes with a few examples, and then the network is trained and tested. Finally, the results of the second experiment are discussed, and future work is debated.

The sections of this paper are organized as follows: Next section presents related work, and then section 1 focuses on describing the network architecture, the training procedure and the results of the first experiment. Section 2 describes the data augmentation procedure and the results of the second experiment. Finally, conclusions and further work are presented.

2 Related work

Among the most influential works in the field of computer vision is the work by Krizhevsky et al. [1], who introduced the AlexNet model that achieved a breakthrough in the ImageNet classification task using deep convolutional neural networks. This work laid the foundation for many subsequent developments in deep learning for computer vision applications.

One such advancement is the deep residual learning framework proposed by He et al. [2]. Their ResNet architecture enabled the training of much deeper networks by using skip connections, thereby achieving state-of-the-art performance in CV tasks.

In recent years, transformers have gained significant awareness as an alternative to CNNs for image recognition tasks. Dosovitskiy et al. [3] demonstrated that transformers could perform competitively on image recognition tasks, outperforming CNNs when trained on large datasets.

Considering the image classification task, several NN architectures, like vanilla NN, CNNs or Vision Transformers, can be implemented to accomplish the job. However, vanilla networks have been shown to perform poorly for this task because they need much more connections than a CNN [1] and are not translation invariant. On the other side, although Vision Transformers are showing promising image classification results, their good performance relies on training on enormous datasets [3], which makes them impractical. This is why for this specific problem, a CNN seems the most reasonable choice. The following section describes the architecture of the neural network.

3 Part 1 - Basic DNN Training

3.1 Part 1: Methods

3.1.1 Architecture

The architecture was inspired by the residual network that achieved SOTA results in 2015 [2]. As the images in the GTSRB have different characteristics, the network was designed to process 112x112 RGB images. Some architectural tricks like batch normalization [4] and ReLu activation functions [1] were included to achieve better performance and decrease training time.

Residual Block (ResBlock): This is a building block for the main neural network. Each residual block consists of a series of layers, including convolutional layers, batch normalization, and ReLU activation functions. There are two cases in the ResBlock implementation:

1. When the input and output channels are equal, the input tensor directly contributes to the output, which is the sum of the input and the processed tensor after passing through the layers in the block (identity mapping [2]).
2. When the input and output channels are not equal, we have to use a projection shortcut [2]. The input tensor is transformed using a 1x1 convolutional layer to match the output channel dimensions. The output is the sum of the processed and transformed input tensors.

Main network architecture (Network): it combines multiple residual blocks. It is composed of the following layers:

1. An initial convolutional layer with 3 input channels, 32 output channels, a kernel size of 3, a stride of 2, and padding of 1. This is followed by batch normalization and a ReLU activation function.
2. Six residual blocks (ResBlock) with various input and output channel sizes, resulting in an increase in the number of channels and a reduction in spatial dimensions.
3. An average pooling layer with a kernel size of 2, which further reduces the spatial dimensions.
4. A Flatten layer that reshapes the tensor to a one-dimensional tensor.
5. A fully connected linear layer with 1584 input features and 43 output features, followed by a softmax activation function that converts the logits to probability distributions.

3.1.2 The Dataset

The dataset is divided into two parts: one for training and one for testing. The training data account for 39209 images, while there are 12630 for testing. The training data distribution is presented in Figure 1.

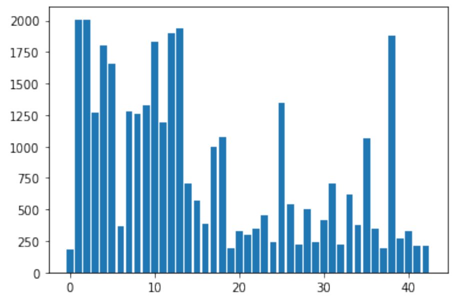

Figure 1: Distribution of the number of images over labels. This distribution accounts for the training data

We can notice that there is a class imbalance that can affect the performance of the model. For example, some classes have 10 times less data than those with more examples. For this first experiment, the data distribution is not changed.

3.1.3 Training procedure

A batch size of 400 is used. The training procedure runs for 40 epochs, and the optimization algorithm used to train is Adam, with a learning rate of 0.0001. A validation set of 10% of the data is set aside to test the model's performance once each epoch. Before the network processes images, they are resized to be 112x112 and normalized.

The loss function used to train the network is the cross-entropy.

3.2 Part 1: Results

The results of the training procedure are illustrated in Figure 2. Between training and validation, the minimum loss reached almost the same value: 2.81 for training and 2.82 for validation. The validation accuracy had a maximum of 98.06%. These results indicate we are not overfitting the data because validation and training are not yet diverging.

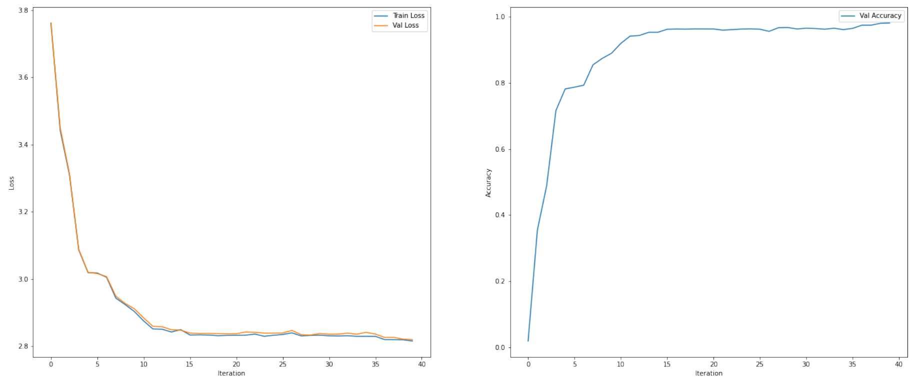

Figure 2: Loss and accuracy performance through the training process

It can be noticed that after 15 epochs, the metrics stabilize, and improvement starts to flatten. However, it is a fact that there is a slow improvement: from epoch 15 to 40, there is almost a 2% of validation accuracy improvement. These could indicate that running for more epochs could result in a slightly better validation accuracy.

Then, the network's performance is evaluated on the test set. It achieves an accuracy of 92.98%, and the confusion matrix is presented in Figure 3. It can be evidenced that the network is entirely wrong when it needs to classify images from classes 27 and 34. One reason for this result could be the small number of images for these two classes in the training dataset. The other reason is that images from these two classes have similar patterns to the ones the network is predicting incorrectly. This means that the network is not able to differentiate those patterns.

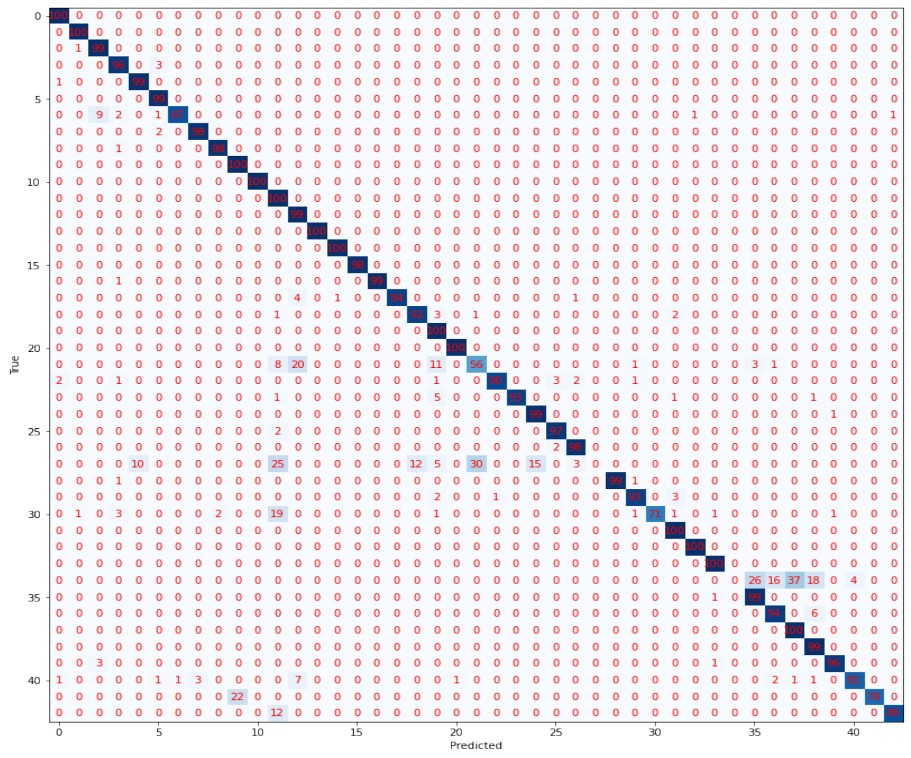

Figure 3: Confusion matrix over the test data. Each row is normalized to sum 100%

3.3 Part 1: Discussion

The model achieves relatively good performance, reaching perfect scores for some classes. Nevertheless, it is confusing the patterns in some classes and biasing towards similar patterns detected in other classes. Several things could be tested to tackle this issue:

1. The training procedure is getting stuck in a local minimum. Therefore, a better-guided training procedure could be implemented to guide the steps towards other directions so the optimization does not end at the same local minimum.
2. The loss function could penalize more when the network confuses these similar patterns.
3. We can assume the network has the capacity to differentiate these similar patterns and just perform data augmentation to upsample the classes with a low number of examples so we reduce the probability that the network gets stuck in a local minimum.

Given that the first two options are less likely to be automated, the third option is a good first approach.

4 Part 2 - Mitigating class imbalance

4.1 Part 2: Methods

Considering the results of part 1, a second experiment is performed in which the same neural network is trained on an augmented training dataset. Then, its performance is assessed on the unmodified test data.

4.1.1 Data augmentation and training procedure

The data augmentation process aims to balance the dataset by increasing the number of samples in classes with fewer samples than a specified cutoff. This is achieved by applying random transformations to the original images, such as color jittering and Gaussian blurring. For each class folder, if the number of samples in that class is below the cutoff, the required number of augmented samples to reach the cutoff is calculated. Then, an image of that class is randomly selected, and a randomly chosen transformation from the list of possible transforms (the two transforms defined previously) is applied. Finally, the augmented image is saved as a new one with the same class label. The cutoff used for this experiment was 1000.

After applying the previous process to the training data, the new distribution is depicted in Figure 4.

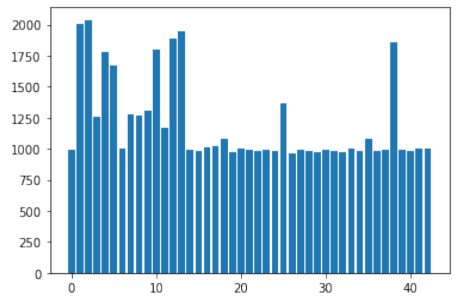

Figure 4: Distribution of the number of images over labels after augmentation. This distribution accounts for the training data

The same parameters used for the first experiments were kept for the training procedure.

4.2 Part 2: Results

The results are illustrated in Figure 5. We can see the same pattern as the first experiment, where between training and validation, the minimum loss reached almost the same value: 2.00 for training and 2.01 for validation. The validation accuracy had a maximum of 99.93%. With this new approach, the validation's loss and accuracy improved, and we can notice that the performance stays the same after 15 epochs, meaning that it converges faster than the first approach.

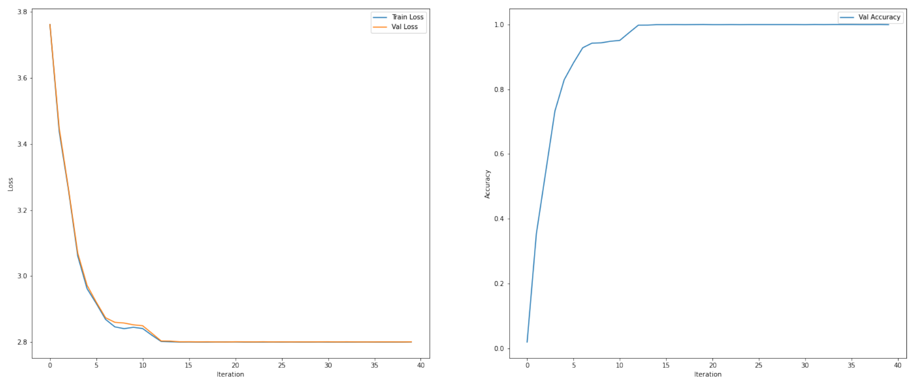

Figure 5: Loss and accuracy performance through the training process

The network's performance on the test set has an accuracy of 95.62%, and the confusion matrix is presented in figure 6 [fig6]. This approach achieved roughly a 3% increase in accuracy concerning the first experiment. Additionally, the confusion matrix shows a much nicer performance. The model is able to differentiate the patterns that troubled it in the first experiment. Nevertheless, it still struggles with images from classes 21 and 22. For class 21, the performance is better than the first experiment. It goes up from 56% to 63%. However, for class 22, the performance is lower by 5%, achieving just 85%. These two classes were in the group that was upsampled in the data augmentation procedure.

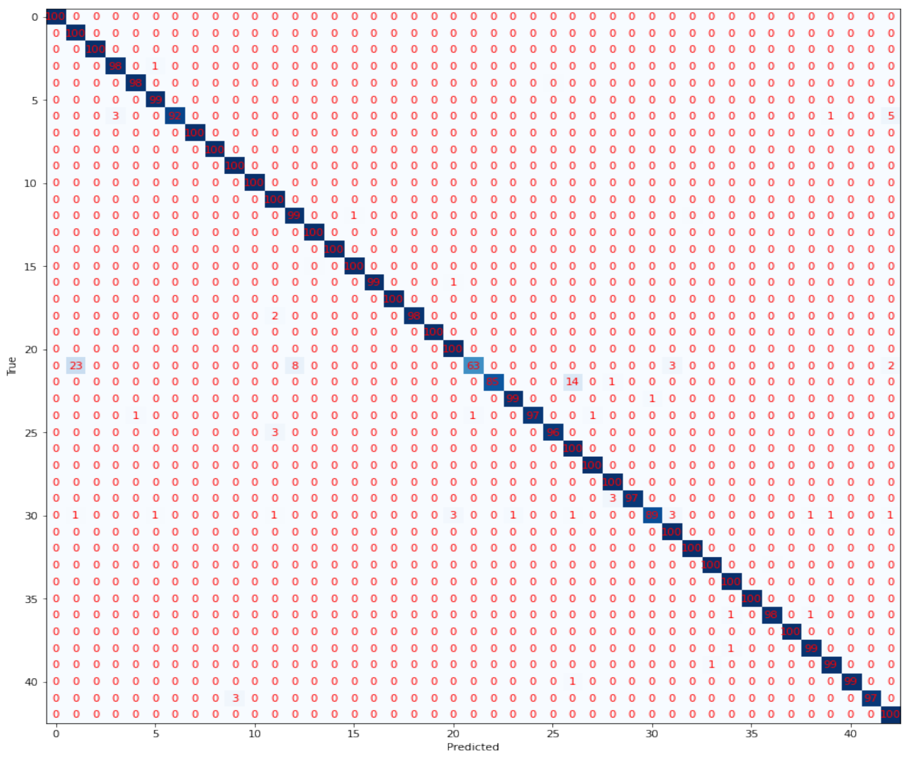
Figure 6: Confusion matrix over the test data. Each row is normalized to sum 100%

4.3 Part 2: Discussion

This new approach showed significant improvements in the model's performance. In addition, augmenting the number of images resulted in an efficient method that could be implemented without changing the architecture or the optimization method.

The results of this last experiment showed a plateau in the performance of this network. The validation accuracy reached almost a perfect score (99.9%), meaning that more training time will not enhance performance. To reach a better performance, some of the following experiments could be performed:

1. Guide the data augmentation process to focus more on the classes with lower accuracy. Nevertheless, when we use the test data to guide the training process, we are biasing the model, so in the end, we are using test data to fit the training.
2. Refine the data augmentation process with more transformations so the training data has more variance and complexity.
3. Make a deeper network architecture enabling it to catch more complex patterns.
4. Try regularization through dropout.
5. Define a pre-training stage with other datasets where the network can catch other patterns not present in the training set so it is less prone to bias.

5 Conclusion

In this paper, we presented a ResNet-based convolutional neural network for traffic sign classification using the German Traffic Sign Recognition Benchmark (GTSRB) dataset. Our proposed model effectively tackled the challenges posed by the diverse set of traffic sign images and the class imbalance present in the dataset.

The data augmentation techniques used in this study not only helped balance the dataset but also enhanced the model's ability to generalize to new samples, making it a robust solution.

While the model achieved significant improvements in performance, there are still opportunities for further enhancements. Future work could explore refining the data augmentation process, deeper network architectures, regularization techniques such as dropout, and pre-training with other datasets.

References

[1] Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[2] He, K., Zhang, X., Ren, S. & Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 770-778).

[3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In Proceedings of the International Conference on Learning Representations (ICLR).

[4] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Proceedings of the 32nd International Conference on Machine Learning (ICML) (pp. 448-456).

[5] LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.