



# Documentation of the program design

## Digital image processing and pattern recognition

Course of study Electrical engineering

Field of study Vehicle electronics

Baden-Württemberg Cooperative State University Ravensburg, Campus Friedrichshafen

by

Kaan Aydin

Submission date:	23rd June, 2024
Processing period:	04/01/2024 - 06/23/2024
Registration number:	9752327
Course:	TFE21-2
Company:	ZF Friedrichshafen AG
Course lecturer:	Jonathan Spieler
Director of Studies of the Dual University:	Prof. Dr.-Ing. Thomas Kibler

# 1 Introduction

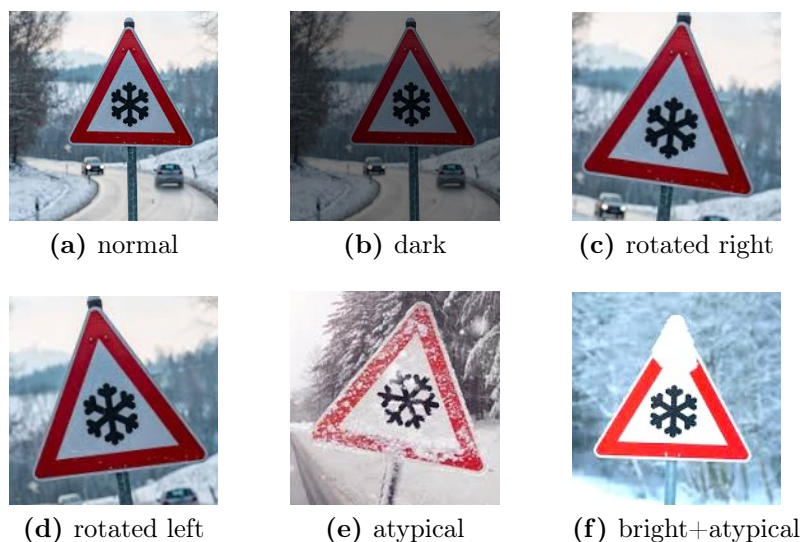
In the realm of autonomous driving (AD) and advanced driver-assistance systems (ADAS), the ability to accurately and efficiently recognize traffic signs is paramount. Traffic signs provide critical information about road conditions, regulations, and warnings, ensuring the safety and smooth operation of vehicles. Neural networks have shown significant promise in addressing this complex task due to their ability to learn and generalize from large datasets.

Traffic sign classification involves identifying and categorizing various traffic signs captured through cameras on the vehicles. This task presents several challenges, including variations in sign appearance due to lighting conditions, weather, occlusions, and the physical state of the signs (e.g., damaged or dirty signs). Traditional image processing techniques often struggle to cope with these variances, leading to the rise of neural network-based approaches.

This documentation provides a comprehensive overview of the development and implementation of a neural network model for traffic sign classification. We will explore the safety evaluation batch, the evaluation metrics, the determination of robustness and epistemic uncertainty. The goal is to offer a clear understanding of how neural networks can be effectively utilized to enhance the reliability and safety of modern vehicular systems through accurate traffic sign recognition [1].

## 2 Safety Evaluation Batch

The batch folder is an essential component in the workflow of training a neural network for traffic sign classification. It contains all the batches of data that are fed into the neural network during the training process. Batches are subsets of the training dataset, used to update the model parameters iteratively. Storing the data in batches helps in managing memory usage and enables faster processing by leveraging parallel computation. Each batch typically contains a balanced mix of various traffic sign images, ensuring that the model learns to generalize well across different sign types. Some of these pictures were taken by hand or used on the Internet. In total, this folder contains 552 images of traffic signs with different variances in brightness, rotation, or type, as shown in Figure 1.



**Abbildung 1:** Extract of a class from the evaluation batch (folder 30)

## 3 Evaluation Metrics

Evaluating the performance of a neural network model for traffic sign classification involves several key metrics that provide a comprehensive understanding of the model’s capabilities and areas for improvement. In this documentation, we focus on four primary metrics: accuracy, precision, specificity, and F1-score. These metrics are essential for assessing the model’s effectiveness in recognizing and classifying traffic signs accurately.

### 3.1 Accuracy

Accuracy is the ratio of correctly classified instances to the total number of instances. It provides a general measure of how often the model correctly predicts the traffic sign category. Accuracy is calculated as:

$$Accuracy = \frac{True\ Positives + True\ Negatives}{Total\ Instances} = \frac{T_P + T_N}{I_{Total}} \quad (1)$$

In traffic sign classification, a high accuracy indicates that the model correctly identifies a significant portion of the signs. However, accuracy alone can be misleading, especially if the dataset is imbalanced.

## 3.2 Precision

Precision is the ratio of correctly predicted positive instances to the total predicted positives. It measures the model’s exactness or the proportion of true positive predictions out of all positive predictions. Precision is defined as:

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Negatives} = \frac{T_P}{T_P + F_N} \quad (2)$$

High precision is crucial in scenarios where false positives are particularly problematic. In the context of traffic sign classification, high precision ensures that the signs identified by the model as belonging to a particular category are indeed correct, minimizing the risk of misclassification.

## 3.3 Specificity

Specificity (also known as the true negative rate) measures the proportion of actual negatives that are correctly identified as such. It indicates how well the model can identify negative instances. Specificity is calculated as:

$$Specificity = \frac{True\ Negatives}{True\ Negatives + False\ Positives} = \frac{T_N}{T_N + F_P} \quad (3)$$

In traffic sign classification, high specificity means the model effectively avoids incorrectly classifying non-target signs as target signs. This is particularly important for ensuring the model does not falsely recognize irrelevant objects as traffic signs, which could lead to incorrect actions in an autonomous driving system.

### 3.4 F1-Score

F1-score is the harmonic mean of precision and recall (recall is the ratio of correctly predicted positive instances to all actual positives). The F1-score provides a balance between precision and recall, offering a single metric that accounts for both false positives and false negatives. It is calculated as:

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (4)$$

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives} \quad (5)$$

$$\rightarrow F1 - Score = \frac{2T_P}{2T_P + F_P + F_N} \quad (6)$$

The F1-score is especially useful for evaluating models on imbalanced datasets, where it is important to consider both precision and recall. A high F1-score indicates that the model has a good balance of correctly identifying traffic signs while minimizing incorrect classifications [2; 3].

By employing these evaluation metrics we can thoroughly assess the performance of our neural network model for traffic sign classification. Each metric provides unique insights into different aspects of the model's performance, ensuring a well-rounded evaluation. These metrics help in fine-tuning the model, comparing different models, and ultimately deploying a robust and reliable traffic sign recognition system for autonomous driving applications.

## 4 Determination of Robustness and Epistemic Uncertainty

When developing a neural network model for traffic sign classification, understanding and quantifying robustness and epistemic uncertainty is crucial for ensuring reliable and safe performance in real-world scenarios. Here, we describe how these concepts are determined and their importance in the context of traffic sign classification.

### 4.1 Robustness

Robustness refers to the model's ability to maintain high performance under varying and potentially adverse conditions. For traffic sign classification, robustness is determined by testing the model against various perturbations and challenging scenarios that it might encounter in real-world driving conditions [4]. The following are included in:

- **Data Augmentation:** The `'load_and_transform_data'` function applies various data augmentation techniques such as random rotation, color jittering, and Gaussian blurring to the training data. This helps the model generalize better data.
- **Evaluation Metrics:** The code logs various evaluation metrics such as accuracy, precision, specificity, and F1-score, which provides a comprehensive understanding of the model's performance.

The robustness of the model can be improved by modifying the training dataset through various transformations. During the training phase, the model is exposed to challenging conditions, including rotations and fluctuations in brightness. This approach helps the model learn to perform reliably even when the input data differs from the ideal or expected conditions. By training under these difficult scenarios, the model becomes more adaptable and less likely to fail when faced with unexpected variations [5].

The calculation method to determine the robustness score is documented in the notebook below the code. Anyway, the value of the robustness-score is 11.4%. This value indicates that the model is 11.4% robust, which is relatively low. It means that the model is resilient to perturbations or adversarial attacks only 11.4% of the time and can still make correct predictions. It is necessary to improve the model to make it more resilient to adversarial attacks or data noise. This could be achieved through various techniques, such as data augmentation, adversarial training, regularization techniques, or more robust model architectures.

## 4.2 Epistemic Uncertainty

Epistemic Uncertainty refers to the uncertainty in the model parameters due to limited knowledge or data. It captures the model's confidence in its predictions and is crucial for understanding when the model is likely to make incorrect predictions. For example, using the Monte-Carlo Dropout (MCD). The MCD is a technique used to estimate the uncertainty of a neural network's predictions. It involves multiple forward passes through the network with dropout enabled, and then calculating the mean and variance of the outputs [6; 7]. In the provided code in `'eval.py'`, the `'get_monte_carlo_predictions'` generates Monte Carlo samples and uncertainty estimates for a given model by performing multiple forward passes on a dataset. The following values were used for the prediction: `'num_forward_passes = 50'`, `'num_classes = 43'` and `'num_samples = 100'`.

The result was a mean epistemic uncertainty of  $0.0233 = 2.33\%$  and a variance epistemic uncertainty of  $0.0034 = 0.34\%$ .



- **Mean epistemic uncertainty: 0.0233**

Epistemic uncertainty: This form of uncertainty arises from the lack of knowledge in the model. It indicates how confident the model is about its predictions. A low value (such as 0.0233) means that the model is relatively confident in its predictions. The model believes that it has been sufficiently trained and that it has a good understanding of the underlying data distribution.

- **Variance epistemic uncertainty: 0.0034**

Epistemic uncertainty variance: Variance indicates how consistent the model is across multiple predictions. A low variance (such as 0.0034) means that the uncertainty in the model's predictions across multiple runs is low. This suggests that the predictions are stable and do not change much when running the Monte Carlo dropout multiple times.

These values indicate that the model is both confident in its predictions and consistent across different predictions. This is a positive sign and suggests that the model is well-trained and able to understand the underlying data distribution well. It indicates good generalization ability, meaning that the model is likely to perform well on new, unknown data as well.

Additionally, the position of the code has a big impact. The order of calling the function `'get_monte_carlo_predictions'` after loading the model and before the evaluation is based on the purpose of using Monte Carlo Dropout for uncertainty estimation. By following this order, you can first estimate the model's uncertainty and then evaluate its performance. This allows you to better understand the model's behavior and confidence in its predictions, especially when dealing with new, unseen data. Additionally, calculating the uncertainty before the evaluation allows you to use the uncertainty estimates during the evaluation process if needed. For example, you might want to consider the uncertainty estimates when computing evaluation metrics or when filtering out uncertain predictions. By having the uncertainty estimates available during the evaluation, you can gain more insights into the model's performance and its ability to handle different types of data.

## 5 Conclusion

The evaluation of a model is based on several metrics, including accuracy, robustness, and the number of correctly classified images. Here, the accuracy in the case is 14.1% after the transformation with a robustness value of 11.4%. Similarly, only 63 out of 446 images were correctly classified. Both the low accuracy and the low robustness indicate that the model is currently not working well. There is considerable room for improvement. The model may need to be retrained, provided with more or better data, or designed with a different model architecture.

Overall, it can therefore be said that the model currently shows very low performance with low accuracy and robustness. Targeted improvements in the areas of data quality, model architecture, and training procedures can significantly improve performance. It is important to address these areas systematically and to continuously monitor and evaluate the results. Furthermore, In the appendix, pictures show the execution of the training of the neural network in figure 2 and the results based on the metric output in figure 3.

# Literatur

- (1) Tabernik, Domen, and Danijel Skočaj. “Deep learning for large-scale traffic-sign detection and recognition.” *IEEE transactions on intelligent transportation systems* 21.4 (2019): 1427-1440.
- (2) Scikit-learn. “Precision-Recall“. Retrieved on June 13, 2024 from [https://scikit-learn.org/stable/auto\\_examples/model\\_selection/plot\\_precision\\_recall.html](https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html).
- (3) Artemoppermann. “Accuracy, Precision, Recall, F1-Score und Specificity“. Retrieved on June 13, 2024 from <https://artemoppermann.com/de/accuracy-precision-recall-f1-score-und-specificity/>.
- (4) Sadaf Gulshad, Ivan Sosnovik, Arnold Smeulders (2021): “Wiggling Weights to Improve the Robustness of Classifiers“, UvA-Bosch Delta Lab, University of Amsterdam, Netherlands.
- (5) Yang, Yao-Yuan, et al. “A closer look at accuracy vs. robustness.” *Advances in neural information processing systems* 33 (2020): 8588-8601.
- (6) Alharbi, Mohammed, and Hassan A. Karimi. “Context-aware sensor uncertainty estimation for autonomous vehicles.” *Vehicles* 3.4 (2021): 721-735.
- (7) MMilanés-Hermosilla, Daily, et al. “Monte carlo dropout for uncertainty estimation and motor imagery classification.” *Sensors* 21.21 (2021): 7241.

# Anhang

```
Anaconda Prompt (miniconda3)
Safety batch: Average loss: 0.0031, Accuracy: 95.4 %, Precision: 98.0 %
Epoch 39: 100%|██████████| 333/333 [00:58<00:00, 5.66 batch/s, accuracy=86.9 %, loss=0.264]
Safety batch: Average loss: 0.0030, Accuracy: 95.2 %, Precision: 98.7 %
Epoch 40: 100%|██████████| 333/333 [00:58<00:00, 5.70 batch/s, accuracy=87.2 %, loss=0.236]
Safety batch: Average loss: 0.0031, Accuracy: 95.5 %, Precision: 97.1 %
Epoch 41: 100%|██████████| 333/333 [00:58<00:00, 5.66 batch/s, accuracy=87.0 %, loss=0.385]
Safety batch: Average loss: 0.0032, Accuracy: 95.8 %, Precision: 96.1 %
Epoch 42: 100%|██████████| 333/333 [00:58<00:00, 5.65 batch/s, accuracy=87.2 %, loss=0.231]
Epoch 00042: reducing learning rate of group 0 to 3.1250e-06.
Safety batch: Average loss: 0.0030, Accuracy: 95.5 %, Precision: 98.0 %
Epoch 43: 100%|██████████| 333/333 [00:58<00:00, 5.67 batch/s, accuracy=87.7 %, loss=0.317]
Safety batch: Average loss: 0.0031, Accuracy: 95.5 %, Precision: 98.3 %
Epoch 44: 100%|██████████| 333/333 [00:59<00:00, 5.63 batch/s, accuracy=87.4 %, loss=0.365]
Safety batch: Average loss: 0.0029, Accuracy: 95.7 %, Precision: 95.8 %
Epoch 45: 100%|██████████| 333/333 [01:22<00:00, 4.04 batch/s, accuracy=87.3 %, loss=0.369]
Safety batch: Average loss: 0.0028, Accuracy: 95.9 %, Precision: 97.4 %
Epoch 46: 100%|██████████| 333/333 [02:12<00:00, 2.52 batch/s, accuracy=87.4 %, loss=0.586]
Safety batch: Average loss: 0.0030, Accuracy: 95.6 %, Precision: 95.8 %
Epoch 47: 100%|██████████| 333/333 [02:08<00:00, 2.60 batch/s, accuracy=87.5 %, loss=0.277]
Safety batch: Average loss: 0.0029, Accuracy: 95.4 %, Precision: 97.3 %
Epoch 48: 100%|██████████| 333/333 [00:59<00:00, 5.61 batch/s, accuracy=87.3 %, loss=0.513]
Epoch 00048: reducing learning rate of group 0 to 1.5625e-06.
Safety batch: Average loss: 0.0030, Accuracy: 95.7 %, Precision: 96.4 %
Epoch 49: 100%|██████████| 333/333 [00:59<00:00, 5.63 batch/s, accuracy=87.5 %, loss=0.428]
Safety batch: Average loss: 0.0028, Accuracy: 95.7 %, Precision: 99.3 %
Epoch 50: 100%|██████████| 333/333 [00:59<00:00, 5.63 batch/s, accuracy=87.5 %, loss=0.328]
Safety batch: Average loss: 0.0030, Accuracy: 95.6 %, Precision: 97.4 %
Registered model 'gtsrb' already exists. Creating a new version of this model...
Created version '6' of model 'gtsrb'.

(deepsafety_env) C:\DeepSafety\Abgabe>
```

**Abbildung 2:** Representation of the training of the neural network

## ▼ Metrics (9)










Name	Value
evaluation accuracy 	0.7536231884057971
evaluation loss 	0.020128864538518414
training accuracy 	0.8747653903903904
training loss 	0.006631977508661222
validation F1-score 	0.9866220735785953
validation accuracy 	0.9557057057057057
validation loss 	0.0030048144750531997
validation specificity 	0.7037037037037037
validation_precision 	0.9735973597359736

Abbildung 3: Representation of the metrics of the training session.