Enhancing Traffic Sign Detection: Deep Learning and Synthetic Data Augmentation for the Dutch Context

Roman Nekrasov, Andy Huang & Huub Van de Voort
Jheronimus Academy of Data Science

Abstract

Traffic signs are crucial for road safety, yet maintaining accurate inventories remains challenging due to regional variations and new placements. In the Netherlands, the absence of a standardized public dataset further complicates this issue. This study aims to fill this gap by creating a synthetic dataset using traffic sign templates and extensive data augmentations. Through a robust pipeline of experiments, we optimized our model and achieved an overall accuracy of 82.1%, demonstrating the effectiveness of synthetic data in handling varying sign appearances, lighting conditions, and types of sign degradation. This study underscores the importance of region-specific datasets and highlights the potential of synthetic data augmentation techniques to address regional variations. Future research should continue to refine these strategies to ensure deep learning models remain adaptable to diverse real-world traffic scenarios.

Introduction

Traffic signs are important for road safety, guiding drivers and informing them about road conditions, regulations, and hazards. The provincial government is tasked with road sign maintenance, which is a complex process. Recognizing the importance of accurate and up-to-date traffic sign information, the national government of the Netherlands has taken a pioneering step by introducing a digital inventory of traffic signs (Rijksoverheid 2020). This initiative aims to provide a comprehensive database of traffic sign data, including sign type, exact location, and other relevant details, though it does not include the actual images of the signs. This database is intended to improve the efficiency of asset management by the local government bodies such as municipalities and provinces.

Despite these technological advancements, there are still challenges in achieving a comprehensive inclusion of all traffic signs within the digital registry. Asset management of these signs is the responsibility of the municipalities and provinces for their respective area. When the inventorying was performed in 2020, the municipalities and provinces were not prepared and only introduced this new system in their workflow recently. This created a gap of new placed signs that are not in the current digital registry. Another issue with the current inventory is that it only contains general road signs and to give an example, no signposts or bicycle route signs. This also results in gaps in the registry, complicating the maintenance of an accurate and comprehensive record. This national inventory is only half a solution for proper asset management and there is expressed need of a new inventory of all types of road signs. While a lot of research has been done in the domain of traffic sign recognition, and benchmark datasets for traffic signs available for countries like Germany and Belgium (Timofte, Zimmermann, and Van Gool 2014) (Houben et al. 2013), there is a notable absence of such a dataset for the Netherlands. Traffic signs can vary significantly between regions in terms of design, language, and even color, highlighting the need for region-specific datasets to ensure the effectiveness of traffic sign detection systems in diverse real-world applications.

To bridge this gap, we conducted a study in collaboration with the province of Noord-Brabant, focusing initially on provincial roads with as goal to detect all traffic signs. This includes bicycle routes, hectometer signs and all other types of varying road signs. We extracted high-resolution 360-degree panoramic images from a data collection partner of the Province, Cyclomedia (Cyclomedia 2009), as our primary test data.

Due to the lack of a Dutch road sign dataset, we employed the Grounding DINO model for zero shot object detection on the panoramic imagery (Liu et al. 2023). This model integrates a Transformer-based detector with grounded pre-training, making it proficient at open-set object detection by combining language and visual data to identify non-predefined objects. By applying this model, we automatically generated bounding boxes around detected road signs. This approach significantly contributed to the process of constructing a real-world dataset, lowering the amount of manual labor required.

Still, the test data collection process proved to be resource-intensive, we considered a different approach for

developing the training set. Being inspired by the work of Araar et al. 2020, we opted to synthetically generate these datasets instead. We stored the templates of the traffic signs identified on the provincial roads and applied augmentations to them. This methodology enabled us to generate training data that resembles the real world to a large extend. The implemented augmentations cover variations in sign appearance due to different lighting conditions, angles, and wear and tear. Next to improving the robustness of our detection model the methodology also aligns with the need for developing more reliable traffic sign recognition models that can function optimally even under degraded conditions (Flores-Calero et al. 2024).

This research includes following contributions:

- First, we evaluate the performance of a synthetically trained traffic sign classifier based on the architecture as proposed by Araar et al. 2020 on a manually curated dataset of real-world Dutch traffic signs, where we achieved an accuracy of 82.1 percent.
- Second, we explore the requirements for our development dataset to be representative for the high-resolution panoramic images. This involves conducting experiments with differently augmented versions of our training set. We found that the employed augmentations are effective for all sign types and not just common ones.

Although our research focuses on Dutch traffic signs, it highlights critical global issues such as the need for adaptability in traffic sign detection systems to account for geographical variations (Flores-Calero et al. 2024). By addressing these challenges, our findings offer a valuable starting point for regions with varying traffic sign conventions to develop similar studies that consider local sign designs, language differences, and regulatory contexts.

The remainder of the report is organized as follows: the Background section reviews existing literature on traffic sign detection. The Study Definition section explains the rationale for using Cyclomedia and details the process of obtaining traffic sign cutouts. The DL Modeling Technique Overview describes the deep learning architectures employed and our strategies in improving our model. The Discussion section reflects on the study outcomes, while the Conclusion summarizes the findings and explores the requirements for creating a representative development dataset using high-resolution panoramic images from Cyclomedia, offering practical insights for future research.

Background

A considerable amount of research has been conducted on traffic sign detection using deep learning models, particularly those in the YOLO (You Only Look Once) family, which are renowned for their speed and efficiency in real-time object detection. For example, Khan et al. 2020 applied YOLOv3 to the Korean Traffic Sign Dataset (KTSD) and the German Traffic Sign Detection Benchmark (GTSDB), achieving an impressive 100% recognition performance on the latter. This improvement in performance is quantified through the Mean Average Precision (mAP), with the Advanced Traffic Sign Recognition (ATSR) system enhancing MAP by 8.08%, surpassing the previous best of 90.07%. However, a significant drawback of employing YOLO is the extensive requirement for each image in the dataset to be carefully prepared with labeled bounding boxes around the objects of interest, a process that can be both time-consuming and resource-intensive.

Additionally, a literature review by Flores-Calero et al. 2024 highlighted the prevalent use of YOLO models in traffic sign detection and their performance but also raised concerns about their reliability. Many studies rely on a single statistical summary metric, such as mAP, to measure model performance. This may not fully capture their effectiveness in real-world scenarios. Another significant challenge noted is the fluctuation in lighting conditions within road environments. These fluctuations arise from varying illumination levels on roadways, which directly affect the visibility of traffic signs. In low light conditions, signs may become less visible, increasing the potential for accidents, while in very bright conditions, such as direct sunlight or exposure to headlights from other vehicles, glare can make it difficult for road users to accurately see and interpret signs. Such variability in lighting can also produce unpredictable shadows and contrasts on signs, obscuring essential details and complicating their identification. Furthermore, sign degradation from wear and damage can further hinder the accurate detection and categorization of signs.

Beyond the YOLO family of models, various CNN architectures have also been explored. Natarajan, Annamraju, and Baradkar 2018 introduced a weighted multi-CNN approach, achieving an impressive 99.59% accuracy on the GTSRB dataset. As noted earlier in this paper, this research underscores a persistent challenge in the field: developing models that maintain robustness despite variations in sign appearance caused by differing lighting conditions, viewing angles, and signs' wear and tear. The study demonstrated that accuracy declines significantly with augmentations meant to simulate real-world degradation. Notably, the application of increasingly severe levels of augmentation, such as an extreme Gaussian blur, could result in accuracy reduction as drastic as 77.64%.

An extension of the Faster R-CNN, Mask R-CNN, proposed by Tabernik and Skocaj 2019, has demonstrated exceptionally high performance across various traffic sign categories, particularly those with considerable intra-class variability, achieving an average error rate of only 2-3%. The larger error rates observed in certain problematic categories can largely be attributed to similarities with other categories, wide viewing angles, and significant occlusions.

Subsequent studies have continued to refine and compare the capabilities of different CNN architectures. For exam-

ple, Qin and Yan 2021 conducted a comparative analysis between YOLOv5 and Faster R-CNN, finding that Faster R-CNN not only achieved lower loss but also showed higher precision, as evidenced by consistent trends in precision, recall, and mAP. These findings highlight the ongoing enhancements in CNN models for traffic sign detection.

Study Definition

The absence of a publicly available Dutch Traffic Sign dataset necessitated the collection of our own data. We carefully curated high-quality computer-generated images, ensuring they contained minimal noise.

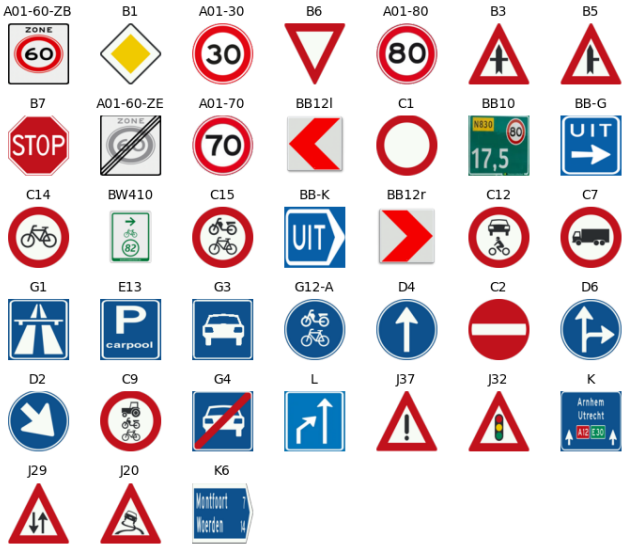


Figure 1: Summary of all the traffic sign class templates used to create the synthetic training data.

From Cyclomedia to Cut-Outs

The next section provides a detailed explanation of the different steps taken in collecting Real World Traffic Sign data. For a visual representation we refer to figure 2. To start, the original Cyclomedia images had a resolution too high for direct use in segmentation. Therefore, we divided the images into overlapping tiles, which were used as input for segmentation. Grounding DINO made predictions on each individual tile by using 'road signs' as a language query parameter. Traffic signs detected across different tiles were then matched using the Intersection over Union algorithm. After gathering the cutouts, we manually labeled them according to official classifications provided by Dutch legislation. This process resulted in 38 traffic sign classes counting 860 images in total.

DL Modelling Technique Overview

In selecting our model architecture, we adopted an approach similar to that used in (Araar et al. 2020), which has demonstrated notable success in predicting traffic signs based on synthetic data. This precedent is crucial given our model's

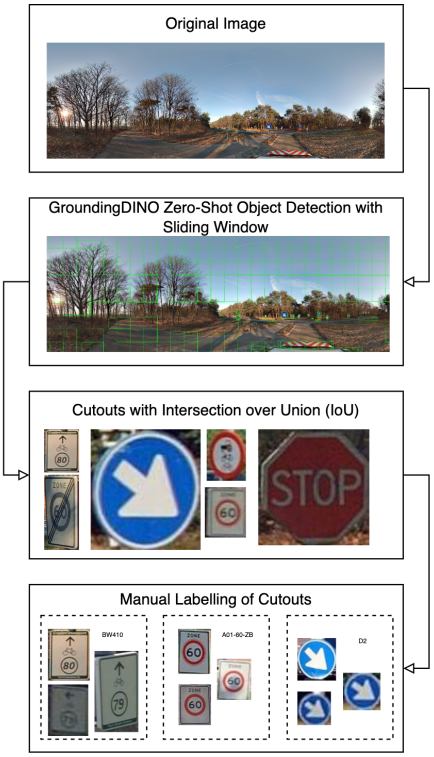


Figure 2: Collection of Real-World Data

requirement to generalize effectively from synthetic to real-world data.

Minor modifications were made to adapt the architecture to our specific needs. Given the high quality and considerable size variation of our traffic sign cutouts in the testing data, we adjusted the dimension of the input layer to 96 by 96 pixels to capture a higher level of image detail.

The architecture of the final model comprises three convolutional layers, two fully connected layers, and an output layer that maps to the 38 different classes. After each convolutional layer, the data undergo normalization and pooling to enhance feature extraction and reduce computational load.

The network's depth is designed to effectively capture the hierarchical patterns inherent in traffic signs. Such depth is crucial, as real-world traffic signs often present in varied lighting conditions, may be obstructed by dirt, or captured from sub-optimal angles.

Network Architecture

The network, designated as `CNN1`, is defined in PyTorch as follows:

- **Convolutional Layers**: The first layer uses 100 filters of size 3x3 with padding to maintain the spatial dimensions post-convolution, ideal for extracting fine-grained features from high-resolution input images. This layer is followed by batch normalization and max pooling, reducing the dimensionality to 48x48 while retaining essential features. Subsequent convolutional layers increase in filter

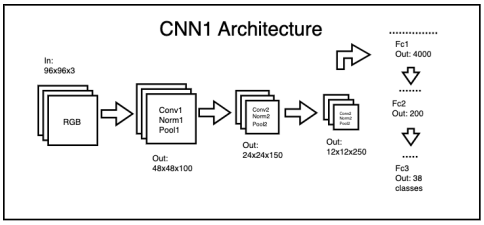


Figure 3: The Architecture of the Convolutional Neural Network

depth, first to 150 and then to 250, allowing the network to construct a progressively richer feature hierarchy.

- **Pooling Layers**: Each convolutional layer is followed by a max pooling layer with a kernel size of 2 and a stride of 2, effectively halving the dimensions and thus reducing the computational complexity while preserving the most salient features.
- **Fully Connected Layers**: Post-convolution, the feature map is flattened and processed through two dense layers with 4000 and 200 units, respectively, which serve to synthesize the features into representations suitable for classification.
- **Output Layer**: The final layer is a linear layer with units equal to the number of classes (38), providing the class scores for each traffic sign.

In the `forward` method, the input tensor `x` is passed sequentially through these layers, with ReLU activations applied to all hidden layers to introduce non-linearity, enhancing the model's ability to learn complex patterns.

Experiments

In this section, we will detail the chronology of the various experiments conducted to improve our model. It is worth mentioning that, the best result obtained from the previously ran experiment is used to run the next experiment. First, we will outline the process of creating our training set with augmentations. As mentioned earlier, we will gradually augment the collected traffic sign templates to build a comprehensive dataset. Each new augmentation will be benchmarked to assess its impact on model performance with the test data, helping us determine whether the training data more accurately represents real-world conditions or if our assumptions are skewed. Additionally, we will describe the specific augmentations applied with different batch sizes and regularization techniques. All experiments were run with Adam optimizer with a learning rate of 0.001. The learning rate was controlled by a scheduler reducing it by a factor of 0.5 every 10 epochs. The model was trained for a maximum of 50 epochs with a patience of 10 on the improvement of the validation loss.

MLOps setup Initially, we experienced trouble setting up a suitable environment for machine learning practices on Google Cloud Platform (GCP). This was due to the GCP credits that we were provided with, did not allow us to create Virtual Machines (VM) with Graphical Processing Units (GPU). We then discovered Vertex AI, where we were able to set up our automated training pipeline using multiple Tesla T4 GPUs. Our models were created using Pytorch, however, we saved the model metrics and our best performing models to cloud storage using Tensorboard.

Synthetic Data Generation

Rotation Traffic signs in real-world scenarios may be oriented or viewed at various angles due to their physical placement or other external factors. By incorporating controlled rotations into our training data, these conditions were simulated, helping the model recognize traffic signs regardless of their orientation (see Figure 4). This strengthens the model's robustness and ensures accurate identification of signs from different angles, which is crucial for real-world performance.

We randomly sample rotation angles from a normal distribution, constrained to a range between -30 and 30 degrees, as more extreme rotations would be unrealistic.

Figure 4: A synthetic rotation from our augmentations which reflects the rotations observed in our test set.

Different Perspectives Building on the fact that traffic signs can be viewed from various angles, multiple perspectives have been integrated into our augmentations. Traffic signs can be seen from the left, right, top, or bottom. To simulate these perspectives, we modify the corner positions to achieve the desired tilt effect (see Figure 5). The degree of tilt is controlled by a factor calculated using the sine of the tilt angle. This angle is randomly sampled from a normal distribution, constrained to a range between -0.5π and 0.5π for side tilts, and between 0 and 1π for forward and backward tilts, allowing us to accurately replicate perspectives from below and above.

Below is described how each tilt type affects the corner positions:

Forward and Backward Tilt:

- Top-left corner: Moves horizontally toward the center of the image based on the tilt angle while keeping the vertical position fixed.
- Top-right corner: Shifts horizontally inward in the opposite direction, maintaining its vertical position.

- Bottom-left corner: Adjusts horizontally toward the center, based on the tilt angle, and shifts upward or downward to simulate respectively the forward or backward tilt.
- Bottom-right corner: Moves horizontally inward like the bottom-left corner and also shifts upward or downward to simulate respectively the forward or backward tilt.

Right Tilt:

- Top-left corner: Remains in its original horizontal position, but moves slightly upwards in its vertical position.
- Top-right corner: Moves horizontally to the left, simulating the sign tilting right, and keeps its vertical position fixed.
- Bottom-left corner: Remains in the same horizontal position, but moves slightly downwards in its vertical position.
- Bottom-right corner: Moves horizontally to the left, aligning with the top-right corner, and maintains its vertical position.

Left Tilt:

- Top-left corner: Moves horizontally to the right while keeping its vertical position.
- Top-right corner: Stays in its original horizontal position but shifts slightly upward in its vertical position.
- Bottom-left corner: Moves horizontally to the right, aligning with the top-left corner, and maintains its vertical position.
- Bottom-right corner: Stays in its original horizontal position but shifts downward to match the proportional tilt.

Figure 5: A synthetic tilt from our augmentations which reflects a tilt observed in our test set.

Background After adjusting the perspectives to simulate different tilts, noisy backgrounds have been added to make the augmented images more challenging and realistic. These noisy backgrounds replicate what we see in the test set, which features varied backgrounds like rough textures, inconsistent lighting, and visual clutter (see Figure 6). We generated the background by creating separate noise channels for red, green, and blue using a normal distribution. These channels are then combined into a three-channel noise background.

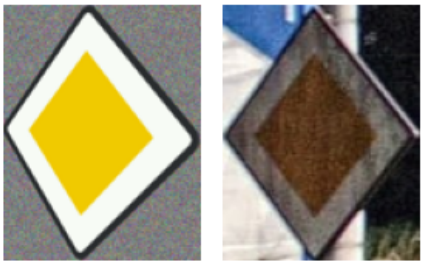


Figure 6: A synthetic background with added noise to the traffic signs which reflects the noisy backgrounds observed in our test set.

Accidental Deformations In addition to adjusting perspectives and adding noisy backgrounds, accidental deformations have been applied to the images (see Figure 7). This was done to reflect the various conditions and degradations that traffic signs experience in the real world. These accidental deformations simulate issues like dents, scratches, fading, and warping, which naturally occur due to weather, aging, or collisions. By incorporating these imperfections, our augmentations better prepare the model to handle signs that may be partially obstructed, distorted, or degraded, making it more robust to the real-world context where signs are often not in pristine condition.

We achieve this by generating a random number of deformation regions. For each region, we randomly select a central point within the image and determine a size within a specified range (a diameter between 5 and 20 pixels). A mask is then created to specify the areas where transparency will be applied. A circular region is drawn on the mask, and the corresponding pixels' alpha values in the image are set to zero, rendering them fully transparent.

Figure 7: A synthetic deformation from our augmentations which reflects the degradations observed in our test set.

Irregular Illumination Thereafter, irregular illuminations have been incorporated to simulate light reflections on traffic signs, which often occur in real-world environments. By adding uneven lighting effects, we can imitate how sunlight or vehicle headlights may reflect off traffic signs, creating bright spots, glares, or shadows (see Figure 8). These lighting anomalies are designed to challenge the model to correctly recognize signs despite inconsistent or

overly bright illumination.

To achieve this effect, we first create an empty illumination layer matching the input image's dimensions. The number of bright spots is sampled from a normal distribution, ranging from 0 to 20. Each bright spot is created by randomly selecting its position, size, and intensity. The position is chosen randomly within the image dimensions, while the size and intensity are determined from specified ranges. Circular spots are then drawn onto the illumination layer with the randomly chosen size and intensity values to replicate realistic lighting reflections. These spots are in turn blurred using a Gaussian filter to soften their edges and give a more natural appearance. Finally, the blurred illumination layer is converted to the same format as the input image and blended with it.

Figure 8: A synthetic illumination from our augmentations, mirroring the light reflections observed in our test set.

Brightness To further augment the dataset, brightness modifications have also been applied to account for varying light conditions in the real world. Depending on the time of day, weather conditions, or surrounding environment, traffic signs can appear darker or brighter. To simulate these variations, we adjust the brightness of our augmented images to reflect changes in natural lighting (see Figure 9).

To achieve this, we convert the image to the HSV (Hue, Saturation, Value) color space so that we can directly manipulate the value channel, which represents brightness. We uniformly sample from a range of -15 to +35 to modify the brightness.

Figure 9: A synthetic brightness level from our augmentations which reflects the varying levels observed in our test set.

Contrast We also modified contrast to reflect the varying visual clarity encountered in the real world. Traffic signs can appear with different contrast levels due to environmental factors like fog, shadows, or glare from sunlight or headlights. By adjusting the contrast, we simulate signs appearing clearer or more washed out (see Figure 10).This is achieved by randomly selecting a factor between 0.5 and 1.5 to adjust the contrast, subsequently OpenCV's cv2.convertScaleAbs function is used to scale the alpha (contrast) with the chosen factor.

Figure 10: A synthetic contrast from our augmentations which reflects the variations in contrast observed in our test set.

Blurs To complete the set of augmentations, we incorporated various blurring techniques because many real-world images are blurred due to different factors such as distance, motion blur, weather conditions and camera quality. Traffic signs in our test set exhibit multiple types of blur, so we used a range of filters and techniques to replicate these effects (see 11):

- **Bilateral Filtering:** This technique smooths the image while preserving edges, which is useful for images with clear details that remain sharp despite noise.
- **Gaussian Blur:** Applies a Gaussian function to blur the image smoothly, simulating a natural out-of-focus effect.
- **Median Blur:** Replaces each pixel value with the median value of neighboring pixels, effectively reducing noise and producing a smoothing effect.
- **Mean Blur:** Averages the values of neighboring pixels to create a more uniform smoothing across the image.
- **Sharpening:** Enhances edges and details to increase clarity, creating a contrast with the blurred regions.

Using these different filters and techniques, we created diverse augmentations that reflect varying blur effects, enhancing the robustness of our model for recognizing traffic signs in challenging visual conditions.

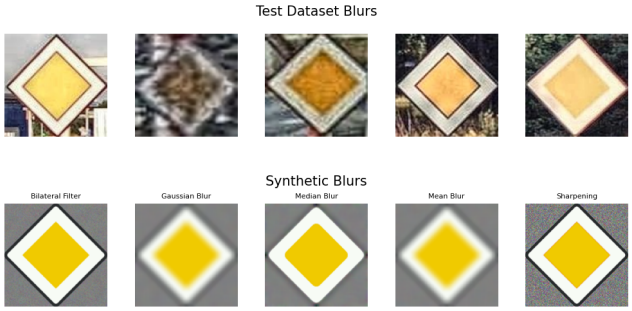


Figure 11: The top row displays various blurs observed in our test dataset, which we have replicated with synthetic blurs (bottom row) using different filters and techniques, including bilateral filtering, Gaussian blur, median blur, mean blur, and sharpening.

Strategy Figure 12 provides a summary of our data augmentation strategy. We begin with a class template of a traffic sign, which undergoes various augmentation stages.

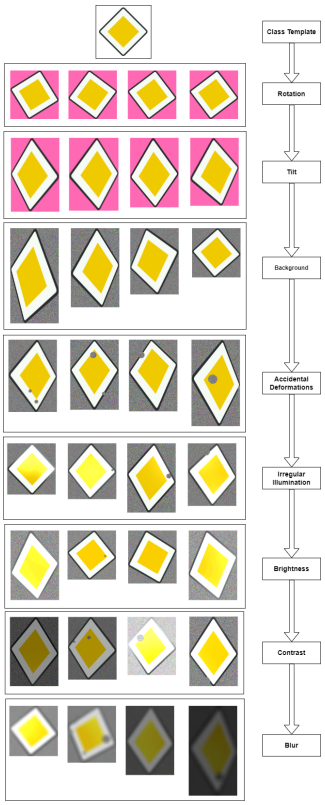


Figure 12: A summary of the strategy used to generate the training data.

At each stage of augmentation, we evaluated the model's accuracy to ensure each modification contributed positively to the final results. The model that utilized all eight augmentations in the full pipeline achieved the best performance (see Figure 13).

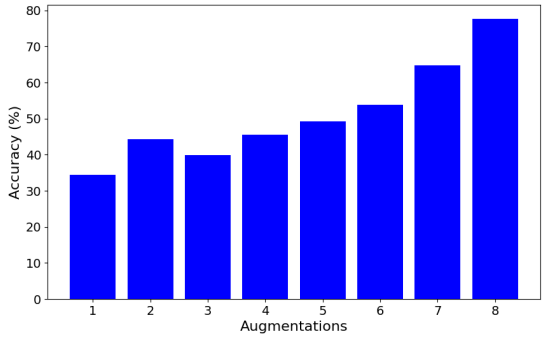


Figure 13: Results of augmentations.

Batch Size

In the case of our training process, batch size is an important parameter. Not only since it influences model training time but also because of its influence on the training process.

Smaller batch sizes tend to introduce more noise into the training data, leading to an irregular pattern in the loss function. This noise can sometimes facilitate generalization, but often results in slower convergence.

Conversely, a larger batch size can cause the loss function to overfit to sharp minimizers, potentially guiding the model towards suboptimal regions of convergence.

Our experiments reveal the importance of finding a good balance between small and large batch sizes. As illustrated in Figure 14, a batch size of 128 yields the highest accuracy, improving performance from 77.7 to 79.1 percent.

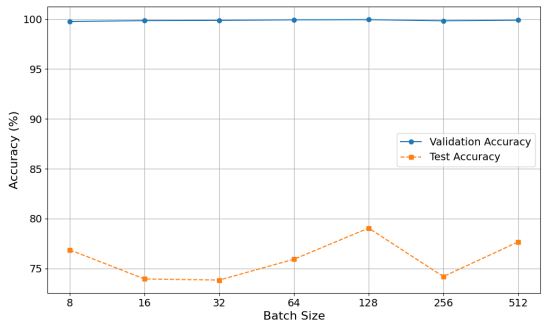


Figure 14: A summary of the strategy used to generate the training data.

Dropout

Including dropout is important for training a traffic sign classifier since the visual nature of traffic signs are prone to overfitting. Adding dropout helps prevent the classifier from relying on a single or small set of neurons, resulting in a classifier that remains resilient against variations in the input data.

Table 1 presents the outcomes of three different experiments, each with varying dropout layer rates. We utilized

dropout rates ranging from 0.1 to 0.5 for the layers following pooling layers and from 0.5 to 0.7 for dropout applied to the fully connected layers.

Configuration	1	2	3
Dropout Pool 1	0.3	0.1	0.2
Dropout Pool 2	0.4	0.2	0.3
Dropout Pool 3	0.5	0.3	0.4
Dropout Fully Connected	0.5	0.6	0.5
Accuracy	**77.2%**	**80.5%**	**82.1%**

Table 1: Dropout rates for the different configurations

L2 Regularization

We implemented explicit regularization by applying a decay rate. This form of regularization penalizes complex model weights, helping to prevent the model from overfitting on the training data.

We experimented with different weight decay factors and achieved the best results with a decay factor of 0.001. This indicates that the regularization effectively balances model complexity by discouraging excessively large weights, resulting in a model that generalizes well to new data.

Decay Factor	0.001	0.005	0.01
Accuracy	**76.9%**	**72.0%**	**70.7%**

Table 2: Accuracy for different decay factors

Discussion

The objective of this study was to develop a traffic sign detection system tailored to the Dutch context by leveraging a synthetic dataset and deep learning-based techniques. This approach directly addressed the gaps in existing national traffic sign inventories, particularly the absence of a standardized dataset of Dutch traffic sign images. With an overall accuracy of 82.1%, the proposed architecture successfully managed the challenges posed by varying sign appearances, lighting conditions, and sign degradations, thereby validating the robustness of our synthetic training data.

Furthermore, by combining the strengths of zero-shot object detection through the Grounding DINO model with comprehensive data augmentation strategies, our approach demonstrated how real-world complexities could be efficiently addressed. Each transformation, from rotations to varying levels of blur, proved crucial in enhancing the model's generalization capabilities across diverse traffic sign conditions. This study not only shows how effective data augmentation can be in creating reliable synthetic datasets, but also highlights the potential of region-specific datasets to resolve challenges that global models may overlook.

Optimizing batch size also proved crucial in preventing overfitting, given the inherent challenges in recognizing traffic signs across varying conditions. We experimented with dropout rates ranging from 0.1 to 0.5 for layers following pooling layers, and from 0.5 to 0.7 for the fully connected layers. These optimal dropout configurations ensured the classifier remained resilient against input variations, thus enhancing its generalization capability.

We also implemented explicit regularization through weight decay to penalize overly complex model weights, aiming to prevent the model from overfitting on the training data. Although a decay factor of 0.001 initially appeared optimal, it did not significantly improve the model's generalization.

This result leads to several possible interpretations and opportunities for future research. First, the model may not be substantially overfitting, suggesting that the current architecture and data augmentation strategies are sufficient for regularization. Most mismatches occurred around the diagonal in the confusion matrix, meaning mismatched signs were often close to the correct class (see Figure 15). However, we acknowledge that the augmentations did not fully reflect real-world conditions (see Figure 16). More realistic augmentations, such as dirty signs and various backgrounds, should be incorporated in future studies.

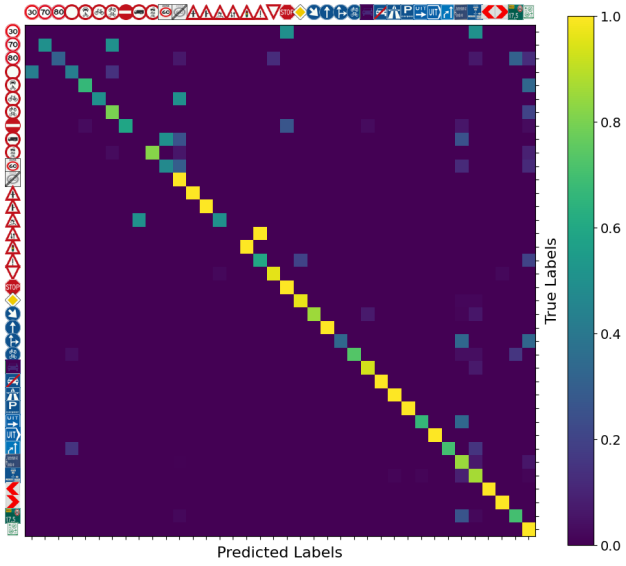


Figure 15: Confusion Matrix of best performing model.

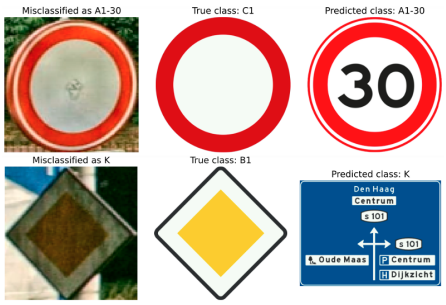


Figure 16: Not included augmentations.

The model's limitations might also result from insufficient or unbalanced data, as some traffic sign classes were underrepresented in the training data due to a lack of template diversity (see Figure 17).

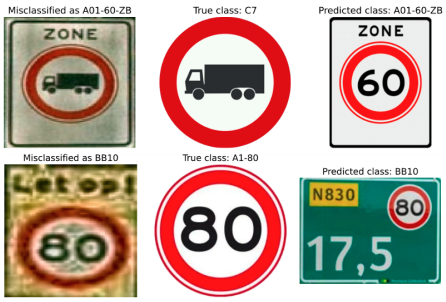


Figure 17: Template under representation.

Furthermore, many low-quality images were included because we considered all traffic signs in the original Cyclomedia images, even those captured from a long distance (see Figure 18). In future studies, one can focus exclusively on higher-quality images, as Cyclomedia captures images approximately every meter, ensuring access to high-resolution images of each traffic sign.

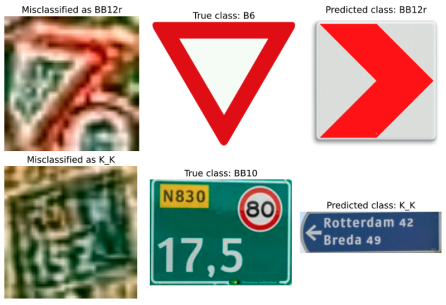


Figure 18: Bad quality images.

Finally, the lack of a standardized Dutch benchmark dataset restricts the generalizability of this study.

Conclusion

In this study, we addressed the challenges of traffic sign detection in the Dutch context by developing a synthetic dataset and using the CNN architecture developed by Araar et al. 2020. By leveraging the Grounding DINO model and comprehensive data augmentation strategies, our approach achieved notable success in recognizing traffic signs with an overall accuracy of 82.1%. This demonstrates the significant potential of synthetic data and augmentation techniques in enhancing model robustness, allowing for reliable generalization across varying traffic sign conditions. Our research highlights the importance of developing region-specific datasets to capture local variations and ensure that deep learning models are more adaptable.

Although weight decay regularization did not yield significant improvements, other strategies like optimal batch size, dropout, and augmentation proved effective in preventing overfitting. The results also indicate that careful consideration should be given to balancing the training data and accurately simulating real-world conditions through realistic augmentations. Future studies should focus on incorporating higher-quality images and refining augmentation techniques to capture more intricate traffic sign variations. Moreover, establishing a standardized Dutch benchmark dataset would improve the generalizability of traffic sign detection systems. Overall, this work lays a valuable foundation for further research in region-specific traffic sign detection, offering practical insights and highlighting the requirements necessary for creating a training dataset representative of real-world images.

References

Araar, O.; Amamra, A.; Abdeldaim, A.; and Vitanov, I. 2020. Traffic sign recognition using a synthetic data training approach. *International Journal on Artificial Intelligence Tools* 29(05):2050013.

Cyclomedia. 2009. Cyclomedia — cyclomedia.com. https://www.cyclomedia.com/nl. [Accessed 04-05-2024].

Flores-Calero, M.; Astudillo, C. A.; Guevara, D.; Maza, J.; Lita, B. S.; Defaz, B.; Ante, J. S.; Zabala-Blanco, D.; and Armingol Moreno, J. M. 2024. Traffic sign detection and recognition using yolo object detection algorithm: A systematic review. *Mathematics* 12(2).

Houben, S.; Stallkamp, J.; Salmen, J.; Schlipsing, M.; and Igel, C. 2013. Detection of traffic signs in real-world images: The german traffic sign detection benchmark. 1–8.

Khan, J. A.; Chen, Y.; Rehman, Y.; and Shin, H. 2020. Performance enhancement techniques for traffic sign recognition using a deep neural network. *Multimedia Tools and Applications* 79(29–30):20545–20560.

Liu, S.; Zeng, Z.; Ren, T.; Li, F.; Zhang, H.; Yang, J.; Li, C.; Yang, J.; Su, H.; Zhu, J.; and Zhang, L. 2023. Grounding dino: Marrying dino with grounded pre-training for open-set object detection.

Natarajan, S.; Annamraju, A. K.; and Baradkar, C. S. 2018. Traffic sign recognition using weighted multi-

convolutional neural network. *IET Intelligent Transport Systems* 12(10):1396–1405.

Qin, Z., and Yan, W. Q. 2021. Traffic-sign recognition using deep learning. In Nguyen, M.; Yan, W. Q.; and Ho, H., eds., *Geometry and Vision*, 13–25. Cham: Springer International Publishing.

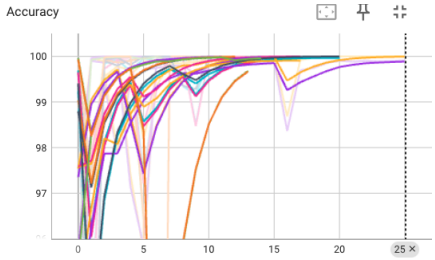
Rijksoverheid. 2020. Verkeersborden in nederland vanaf nu digitaal beschikbaar. Accessed: 2023-05-04.

Tabernik, D., and Skocaj, D. 2019. Deep learning for large-scale traffic-sign detection and recognition. *CoRR* abs/1904.00649.

Timofte, R.; Zimmermann, K.; and Van Gool, L. 2014. Multi-view traffic sign detection, recognition, and 3d localisation. *Machine vision and applications* 25:633–647.

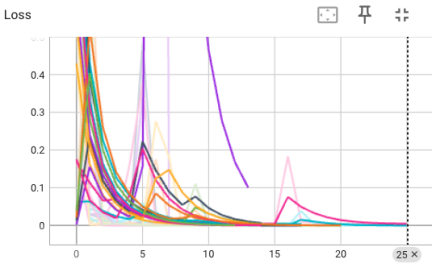
Appendix

Augmentations Experiment Runs



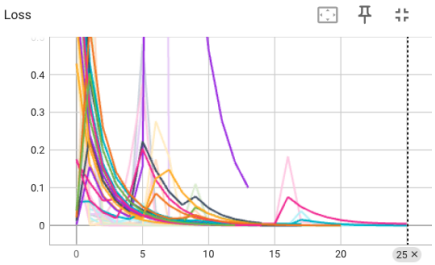
Run	Smoothed	Value	Step	Relative
1/Accuracy_Train	99.6806	100	13	19.78 min
1/Accuracy_Validation	99.6671	99.9825	13	19.78 min
2/Accuracy_Train	99.9786	100	14	21.37 min
2/Accuracy_Validation	99.9596	99.9825	14	21.37 min
3/Accuracy_Train	99.9319	100	14	21.64 min
3/Accuracy_Validation	99.9034	99.9737	14	21.64 min
4/Accuracy_Train	99.9512	99.9825	13	20.13 min
4/Accuracy_Validation	99.9182	99.9386	13	20.13 min
5/Accuracy_Train	99.9918	100	12	18.66 min
5/Accuracy_Validation	99.983	99.9912	12	18.66 min
6/Accuracy_Train	99.9977	100	20	30.87 min
6/Accuracy_Validation	99.9645	99.9649	20	30.87 min
7/Accuracy_Train	99.9972	100	17	26.38 min
7/Accuracy_Validation	99.9087	99.9123	17	26.38 min
8/Accuracy_Train	99.9937	100	25	38.73 min
8/Accuracy_Validation	99.8873	99.9035	25	38.73 min

Figure 19: Augmentations train and validation accuracy.



Run	Smoothed	Value	Step	Relative
1/Loss_Train	0.0068	0	13	19.78 min
1/Loss_Validation	0.1005	0.0005	13	19.78 min
2/Loss_Train	0.001	0.0001	14	21.37 min
2/Loss_Validation	0.0013	0.0005	14	21.37 min
3/Loss_Train	0.0032	0.0002	14	21.64 min
3/Loss_Validation	0.007	0.0012	14	21.64 min
4/Loss_Train	0.0009	0	13	20.13 min
4/Loss_Validation	0.0044	0.0019	13	20.13 min
5/Loss_Train	0.0007	0	12	18.66 min
5/Loss_Validation	0.0013	0.0003	12	18.66 min
6/Loss_Train	0.0003	0	20	30.87 min
6/Loss_Validation	0.0009	0.0007	20	30.87 min
7/Loss_Train	0.0001	0	17	26.38 min
7/Loss_Validation	0.0049	0.0047	17	26.38 min
8/Loss_Train	0.0004	0	25	38.73 min
8/Loss_Validation	0.0051	0.0044	25	38.73 min

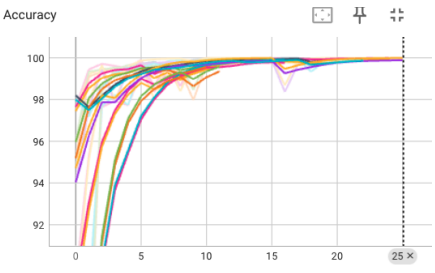
Figure 20: Augmentations train and validation loss.



Run	Smoothed	Value	Step	Relative
1/Loss_Train	0.0068	0	13	19.78 min
1/Loss_Validation	0.1005	0.0005	13	19.78 min
2/Loss_Train	0.001	0.0001	14	21.37 min
2/Loss_Validation	0.0013	0.0005	14	21.37 min
3/Loss_Train	0.0032	0.0002	14	21.64 min
3/Loss_Validation	0.007	0.0012	14	21.64 min
4/Loss_Train	0.0009	0	13	20.13 min
4/Loss_Validation	0.0044	0.0019	13	20.13 min
5/Loss_Train	0.0007	0	12	18.66 min
5/Loss_Validation	0.0013	0.0003	12	18.66 min
6/Loss_Train	0.0003	0	20	30.87 min
6/Loss_Validation	0.0009	0.0007	20	30.87 min
7/Loss_Train	0.0001	0	17	26.38 min
7/Loss_Validation	0.0049	0.0047	17	26.38 min
8/Loss_Train	0.0004	0	25	38.73 min
8/Loss_Validation	0.0051	0.0044	25	38.73 min

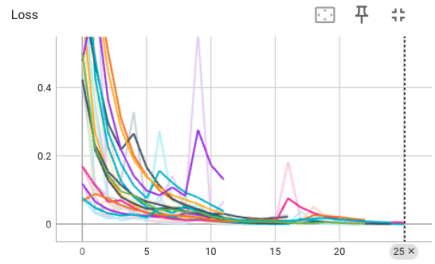
Figure 21: Augmentations test accuracy.

Batch Size Experiment Runs



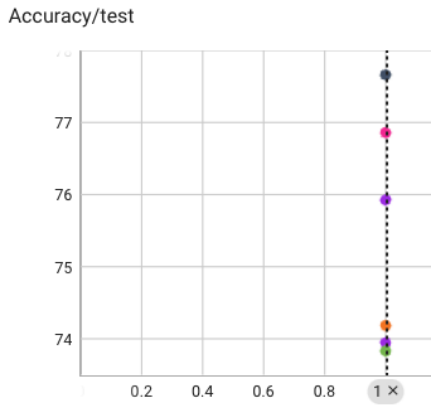
Run	Smoothed	Value	Step	Relative
runs/128/Accuracy_Train	99.9918	100	24	1.446 hr
runs/128/Accuracy_Validation	99.9172	99.9386	24	1.446 hr
runs/16/Accuracy_Train	99.9079	99.9101	16	59.62 min
runs/16/Accuracy_Validation	99.7633	99.7368	16	59.62 min
runs/256/Accuracy_Train	99.7197	99.9956	10	35.56 min
runs/256/Accuracy_Validation	99.5729	99.8421	10	35.56 min
runs/32/Accuracy_Train	99.758	99.9781	11	29.65 min
runs/32/Accuracy_Validation	99.6233	99.886	11	29.65 min
runs/512/Accuracy_Train	99.9937	100	25	39.85 min
runs/512/Accuracy_Validation	99.8873	99.9035	25	39.85 min
runs/64/Accuracy_Train	99.968	99.9978	22	47.01 min
runs/64/Accuracy_Validation	99.8607	99.9035	22	47.01 min
runs/8/Accuracy_Train	99.5756	99.9101	11	55.89 min
runs/8/Accuracy_Validation	99.3506	99.7719	11	55.89 min

Figure 22: Batch size train and validation accuracy.



Run	Smoothed	Value	Step	Relative
runs/128/Loss_Train	0.0007	0	24	1.446 hr
runs/128/Loss_Validation	0.0056	0.0038	24	1.446 hr
runs/16/Loss_Train	0.0067	0.0056	16	59.62 min
runs/16/Loss_Validation	0.022	0.0272	16	59.62 min
runs/256/Loss_Train	0.0146	0.0046	10	35.56 min
runs/256/Loss_Validation	0.0196	0.0071	10	35.56 min
runs/32/Loss_Train	0.02	0.0022	11	29.65 min
runs/32/Loss_Validation	0.0383	0.0094	11	29.65 min
runs/512/Loss_Train	0.0004	0	25	39.85 min
runs/512/Loss_Validation	0.0051	0.0044	25	39.85 min
runs/64/Loss_Train	0.002	0.0002	22	47.01 min
runs/64/Loss_Validation	0.0128	0.0076	22	47.01 min
runs/8/Loss_Train	0.0324	0.0176	11	55.89 min
runs/8/Loss_Validation	0.1309	0.0662	11	55.89 min

Figure 23: Batch size train and validation loss.



Run	Smoothed	Value
runs/128	79.0509	79.0509
runs/16	73.9535	73.9535
runs/256	74.1898	74.1898
runs/32	73.8372	73.8372
runs/512	77.662	77.662
runs/64	75.9302	75.9302
runs/8	76.8605	76.8605

Figure 24: Batch size test accuracy.

Dropout Experiment Runs

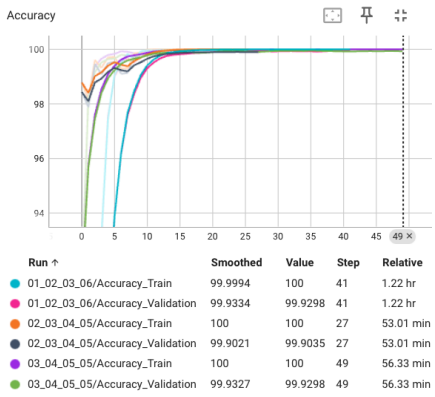


Figure 25: Dropout train and validation accuracy.

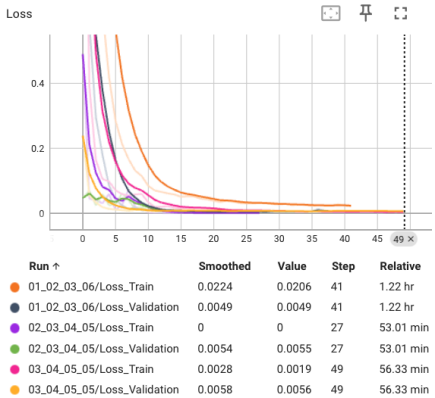


Figure 26: Dropout train and validation loss.

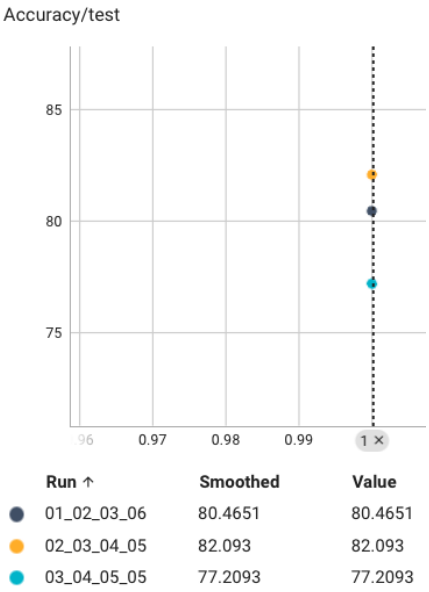


Figure 27: Dropout test accuracy.

L2 Regularization

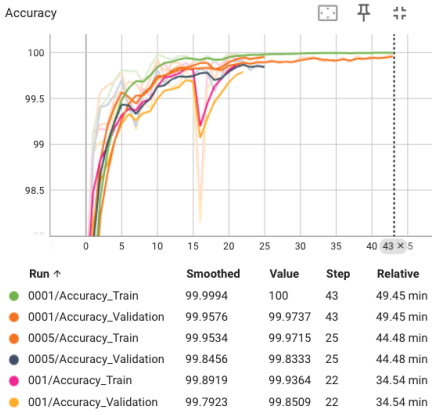


Figure 28: L2 train and validation accuracy.

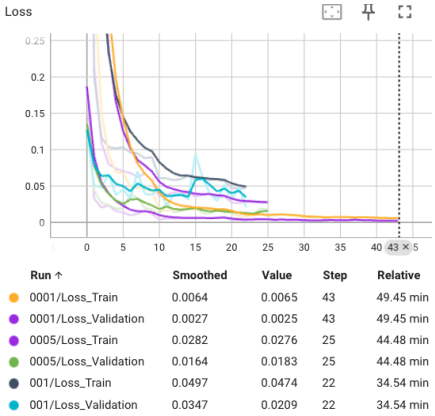


Figure 29: L2 train and validation loss.

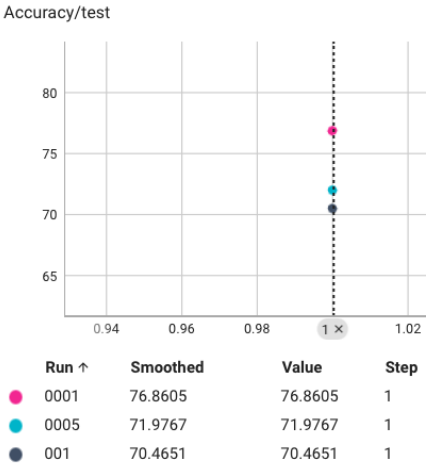


Figure 30: L2 test accuracy.

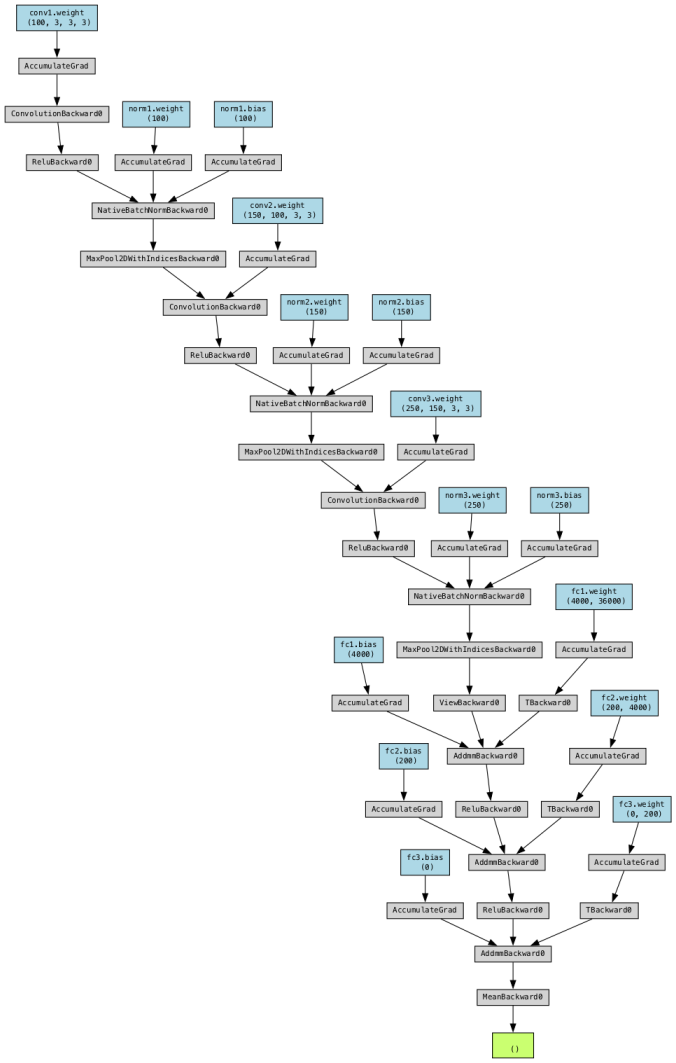


Figure 31: The computational Graph of the CNN Visualized with the Python Torchviz library