machine learning & knowledge extraction

MDPI

Article

Using N-Version Architectures for Railway Segmentation with Deep Neural Networks

Philipp Jaß * and Carsten Thomas

HTW Berlin, University of Applied Sciences, Wilhelminenhofstraße 75A, 12459 Berlin, Germany
* Correspondence: philipp.jass@htw-berlin.de

Abstract: Autonomous trains require reliable and accurate environmental perception to take over safety-critical tasks from the driver. This paper investigates the application of N-version architectures to rail track detection using Deep Neural Networks (DNNs) as a means to improve the safety of machine learning (ML)-enabled perception systems. We combine three different neural network architectures (WCID, VGG16-UNet, MobileNet–SegNet) in a 3M1I configuration. In this configuration, we apply two fusion methods to increase accuracy and to enable error detection: Maximum Confidence Voting (MCV), combining the DNN predictions at the image level, and Pixel Majority Voting (PMV), a novel approach for combining the predictions at the pixel level. In addition, we implement a new method for evaluating and combining prediction confidence values in the N-version architecture during runtime. We adjust the overall prediction confidence according to the conformity of all individual predictions, which is not possible with an individual network. Our results show that the N-version architecture not only enables a detection of erroneous predictions by utilizing those adjusted confidence values, but it can also partially improve the predictions by using the PMV combination algorithm. This work emphasizes the importance of model diversity and appropriate thresholds for an accurate assessment of prediction safety. These approaches can significantly improve the practical applicability of ML-based systems in safety-critical domains such as rail transportation.

Keywords: machine learning; N-version; safety; autonomous rail systems; semantic segmentation; safety-critical AI; model diversity; confidence evaluation; rail track detection

check for updates

Academic Editor: Danial Javaheri

Received: 21 March 2025
Revised: 13 May 2025
Accepted: 18 May 2025
Published: 26 May 2025

Citation: Jaß, P.; Thomas, C. Using N-Version Architectures for Railway Segmentation with Deep Neural Networks. *Mach. Learn. Knowl. Extr.* **2025**, *7*, 49. https://doi.org/10.3390/make7020049

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Machine learning (ML) algorithms are suitable for control tasks that are difficult to formalize, and therefore difficult to master with conventional control approaches. ML approaches are increasingly applied to the control of Cyber–Physical Systems (CPSs) and used in safety-critical contexts in mobility, healthcare, industrial automation, and other fields. One typical application is the use of neural networks for image processing in autonomous driving in the automotive domain, for functions like lane detection and collision avoidance [1]. Nevertheless, the black-box nature of machine learning algorithms hinders the demonstration of safety-related properties of ML-based CPSs, making it difficult to apply the standard processes and methods usually adopted by the safety engineering community [2,3]. Key problems with the use of neural networks for safety-critical tasks relate to traceability issues between requirements and solution elements and characteristics, coverage demonstration issues, and verification issues [4].

In this paper, we present the use of the well-known N-Version programming method from traditional safety engineering for a machine learning-based rail track detection system,

in order to analyze the effectiveness of this method on a real-world application example using machine learning and overcome these difficulties. Rail track detection is one of the core perception functions that is needed for autonomous driving on rails. Due to this function, all further perception tasks, like obstacle detection, are able to distinguish the train's path, i.e., the ego-track, from the rest of the image. The ego-track is the critical area that needs supervision for obstacles of any kind, in order to ensure a safe ride of the train without collisions. Hence, the rail track detection itself is a safety-critical perception task.

The aim of this paper is to investigate the use of an N-version architecture with multiple neural networks for safe rail track detection. Different network architectures are combined in order to detect and, if necessary, correct perception errors. While existing work discusses N-version approaches conceptually, we implement a complete architecture based on real data with quantitative evaluation to detect errors and increase prediction quality. In our work, we create a novel domain-specific algorithm for prediction combination (Pixel Majority Voting) in N-Version architectures, which is analyzed and compared to a classical voting algorithm (Maximum Confidence Voting). In contrast to other approaches, we also develop a novel method for confidence evaluation of DNN predictions, as well as an algorithm to utilize those confidence values for detection of errors in the N-version architecture.

After a brief discussion of related literature, we will present an overview on the N-version approach in Section 3, by explaining all components of the N-version system including the Deep Neural Networks we used for rail track detection and the dataset we used. In the following Section 4, we will present the results of our work, followed by a brief discussion and conclusion in Sections 5 and 6.

2. Related Work

Similar to other industries, the rail sector has also been conducting intensive research into the development of autonomous systems in recent years. The goal is the fully autonomous operation of trains, also known as Grade of Automation (GoA) 4 [5].

Such autonomous trains have the potential to increase both the efficiency and safety of rail transportation. For example, the frequency of human error as a cause of failure can be drastically reduced through the software-based realization of control tasks and the safeguarding of this software through existing or newly created standards and norms [6].

Highly automated or even autonomous trains also enable increased route capacities and support the flexibility of train traffic, especially for regional and freight traffic [7]. These advantages might help to shift traffic from other means of transport to the rails and thus lead to positive effects with regard to climate change.

However, Singh et al. [6] state that there are also considerable technical and regulatory challenges for the development and use of autonomous trains. These include the integration of reliable sensor technologies and the detection and management of unforeseen events.

Precise detection and assessment of obstacles, as well as reliable interaction with other trains and road users, play a decisive role. According to Trentesaux et al., this requires the development of robust sensor technologies and machine learning algorithms, as well as robust safety architectures [8].

Obstacle detection is necessary for highly automated or autonomous trains and is one of many different subjects that are currently being researched in the context of autonomous trains using machine learning methods. In order to reliably classify objects in the vicinity of the train as obstacles, it is necessary to recognize the path of the train. This train path is also known as ego-track [9].

For the reasons mentioned above, rail track detection—the subject of this work—can be regarded as a basic safety-critical function for autonomous driving on rails. A widely

used method for rail track detection is semantic segmentation using convolutional neural networks (CNNs) [10–12]. Li et al. [10] and Risitc-Durrant et al. [13] confirm that the use of CNNs offers advantages over traditional methods that use line or corner features. Traditional methods can be used effectively for individual scenes and, unlike CNNs, do not pose a major hurdle in terms of approval and proof of reliability and safety. On the other hand, they have problems with generalization, especially in the case of dynamic scene changes while the train is moving.

In addition to semantic segmentation, DNN architectures for alternative detection methods, like e.g. key point estimation, have been proposed in the literature [14–16] with promising results.

Other work [17] shows that the combined use of camera and LiDAR sensors for obstacle detection offers advantages. It shows that the use of deep learning techniques in this area enables reliable and fast obstacle detection that even surpasses traditional detection methods in terms of precision and reaction time.

One problem highlighted in the literature is the lack of availability of training data in the railroad sector [13]. Drizi and Boukadoum [18] therefore propose the use of transfer learning and augmentations in the railroad sector as well. Both are widely used techniques in dealing with limited datasets and are used in this work as well.

In addition to all the positive effects that the use of deep learning in the rail sector brings, the use of such algorithms for a safety-critical area such as rail traffic represents a major challenge. This includes, above all, the management of the training and test data used. Ensuring the consistency, accuracy, and completeness of the datasets used is of crucial importance for the generation of reliable neural networks [19].

In addition to the technical challenges mentioned above, there exist a number of regulatory challenges for the use of machine learning methods for safety-critical control tasks. For example, EN 50716:2023 [20] is a core standard for the use of software in the rail sector that defines four safety integrity levels (SILs) (i.e., SIL 1–4), as well as basic integrity (i.e., SIL 0), depending on the criticality of the application area and specifies corresponding safety requirements for the software. However, it currently only highlights the challenges regarding the use of software based on machine learning and does not yet contain any detailed and prescriptive safety requirements for such software. There are currently several standardization initiatives addressing this standardization gap that deal with the definition of standards for ML components, e.g., the standardization roadmap AI in Germany [21].

In order to meet the regulatory challenges for machine learning-based systems, Machida [22] proposes the use of the N-version programming approach. This approach is known in the safety sector. The basic idea is to have a function executed by several instances that are as diverse as possible. By comparing the results, errors in the individual instances can be detected and sometimes even masked. Machida proposes transferring this concept to machine learning-based systems, using multiple neural networks to implement a function instead of a single one. Other authors also support this theory and are researching the positive effects of N-version machine learning [23–25]. As previous results have shown, the increase in reliability that is possible with such an approach depends heavily on the diversity of the individual networks. There are different types of diversity [22], including using different sensors as model inputs, different datasets for model training, or different architectures of the neural networks.

Based on Machida's concepts, we have already proposed a generic architecture for such ML-based N-version systems in an earlier work [26]. Building on this, this paper aims to implement a real N-version system with multiple neural networks for rail track detection using semantic segmentation as an instantiation of this generic architecture and analyzing the results in terms of detection quality and reliability.

Table 1 highlights the limitation of the existing N-version approaches presented, as well as the differences in our own work.

Table 1. Comparison of existing N-version approaches and our own approach.

Criteria	Our Approach	Machida (2019) [22]	Xu et al. (2019) [23]	Wu et al. (2018) [24]	Gujarati et al. (2020) [25]
Aim	Application of N-version ML for safe image segmentation	Theoretical proposal for N-version in ML context	Fault tolerance for DNNs	Resilience of ML-based steering control algorithms	Generalization of NVP principle to ML components
Use Case	Visual rail track detection	None—general safety framework	Image classification, no specific domain	Vehicle control	Image classification and speech recognition, not domain-specific
Level of Implementation	Complete implementation with multiple DNNs and evaluation	No implementation, only concept	Experimental	Implementation with weighted combination	No implementation, only numerical evaluation
Safety Integration	Safety-by-Design with architectural diversity and confidence evaluation	Safety focus, no evaluation	Only robustness evaluation	Focus on reliability, safety not systematically integrated	Safety context emphasized, numerical reliability evaluation

3. N-Version Approach for Semantic Segmentation

Semantic segmentation is a task in computer vision that involves assigning a semantic label to each pixel in an image. This can be useful for applications such as autonomous driving, where it is important to understand the scene and identify objects such as cars, pedestrians, and buildings [27]. In the following section, the concept for an N-version architecture for rail track detection using semantic segmentation will be explained.

3.1. N-Version Architecture

The basic principle of N-version architectures is to use multiple different implementations for the same functionality or task. This allows for error detection and thus increases the reliability and robustness of the overall system. Machida [22] proposes the use of such architectures for systems using neural networks. He proposes different N-version configurations, e.g., double-model single-input (DMSI) or triple-model triple-input (TMTI), and states that a high diversity in both models and sensor inputs allows for higher reliability. In one of our previous works [26], we proposed a generic NMXI architecture based on these variants, which allows almost arbitrary configurations of models and sensor inputs. In addition to the individual neural networks, we also consider different algorithms for combining the individual predictions. In addition to diversity, these algorithms also have a major impact on the performance and potential for error detection in the overall system.

For this work, we implemented a total of three neural networks for rail track detection and integrated them into a N-version architecture. The resulting architecture is shown in Figure 1 (architecture diagram) and is an instantiation of the generic NMXI architecture, which can be described as 3M1I, corresponding to the TMSI (triple-model single-input) from Machida's paper, as all neural networks work on the same input images.

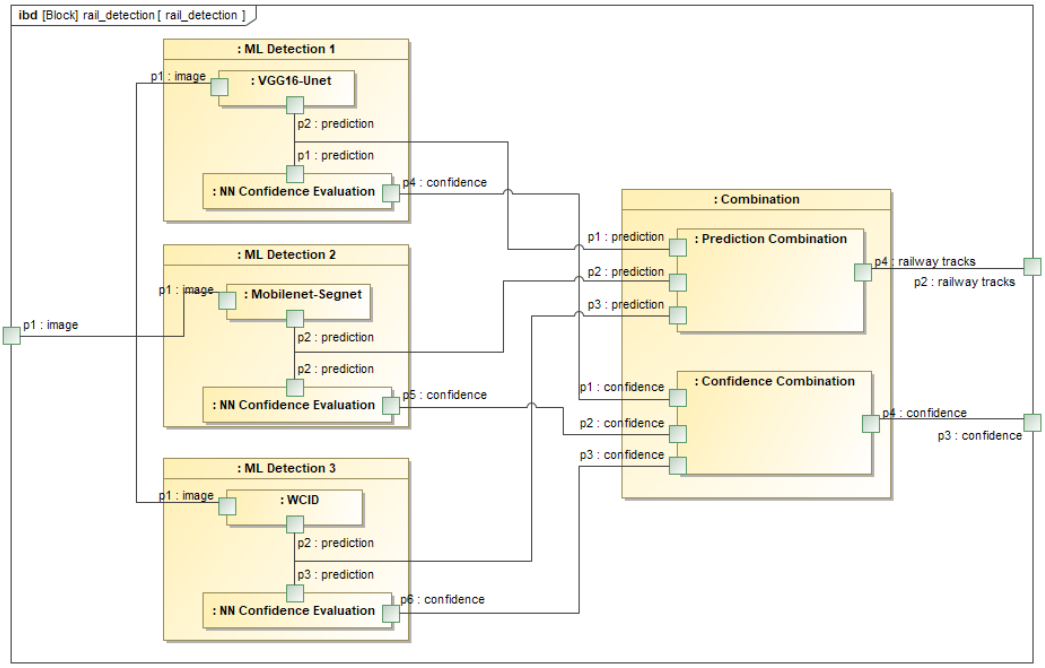


Figure 1. N-version architecture instantiation for rail track detection, used in this paper. It shows the 3M1I architecture pattern, using three DNNs and a combination block.

In addition, a major focus of the N-version architecture presented in this paper is the Confidence Score. This score is calculated for the individual neural networks as well as for the entire system and indicates how well the system estimates its own predictions. This can be seen in the figure by the confidence evaluation blocks that are present for each neural network, as well as the combination block.

Each element of the 3M1I architecture is described in detail in the following sections: the neural networks, the confidence calculation and the combination algorithms, as well as the dataset for training and evaluation.

3.2. Dataset

The basis for the dataset used in this paper is the RailSem19 [28] dataset, which is widely used for railway image segmentation in the literature. This dataset consists of 8,500 RGB images recorded from the train driver's perspective. We included this dataset in our study for the following main reasons:

- It includes scenes from various environments;
- It contains relatively complex scenes (multiple tracks and switches per image);
- It is a free dataset for public use.

The segmentation masks in this dataset contain labels for 21 classes. Of these 21 classes, only 2 are relevant for the rail track detection task. These two classes describe the metal rails (i.e., rail-raised) and the track bed (i.e., rail track). For our training, we combine these two classes into one class, the Track class, and all other classes into the Background class.

The dataset used in this paper is extended by our own dataset, in order to increase the number of images for training, validation and testing. This dataset is based on videos [29] from the Nordlandsbanen (NLB) track in Norway. These videos were recorded for documentation purposes in all four seasons and hence contain various weather and lighting conditions, which makes it a good source for diverse training data. From these recordings, single frames were extracted, for which the segmentation masks were manually created. In this way, 7,019 labeled mask–image pairs were generated. In addition to the track masks, information tags were labeled for each image. The tags describe key properties of the image,

e.g., weather conditions, lighting conditions, time of day, etc., that can be used to study the influence of such situational parameters on the perception performance.

With both previously described datasets combined, a total of 15,519 images were used for training and evaluation. For all images, only the ego-track was considered for the segmentation masks. Thus, the neural networks were directly trained to predict the train's path. The whole dataset was divided into training, validation, and test sets. Table 2 shows the corresponding distribution of the images across these datasets. For the test set, only images from our own dataset were considered. This decision was made because no information tag labels are available for the RailSem19 dataset. These information tags allow for filtering of the predictions on the test set, in order to determine weaknesses and strengths of the neural networks and of the combination algorithms, which is a feature we used in our experiments.

Table 2. Dataset split.

Dataset	Training Set	Validation Set	Test Set
Combined	10,863 (70.0%)	2328 (15%)	2328 (15%)
RailSem19	6966	1534	0
NLB	3897	794	2328

With 15,519 images, the dataset is comparatively small for neural network training on a semantic segmentation task. Due to the size limitations of our dataset, we have to deal with overfitting of the models during training. For this reason, we use dropout layers as well as batch normalization layers in the models and train the models for a limited amount of epochs.

3.3. Rail Track Detection Neural Networks

As mentioned earlier in this paper, a total of three neural networks were implemented for rail track detection. To comply with the N-version approach, all the networks are implemented for the same task–semantic segmentation of RGB camera images. All networks are trained using the dataset described in the last section, with images resized to 1920×1056 pixels.

In the following, all three neural networks are described in detail.

3.3.1. WCID

The first neural network is a version of a fully convolutional network (FCN) for semantic segmentation. This network was developed for lane detection in the automotive domain. It was presented in a paper by Michalke et al. entitled "Where can I drive? A Systems Approach: Deep Ego Corridor Estimation for Robust Automated Driving" [30]. Since the authors do not give an explicit name for their network architecture, it will be referred to as WCID (short for "where can I drive?") in this paper.

This network is, compared to the other segmentation networks, a small and lightweight architecture. It has approximately 500,000 trainable weights. We use this network in our experiments to show the influence of such small and limited architectures on the prediction combinations and the overall prediction, as well as the confidence scores. Like most segmentation networks, the WCID network consists of an encoder for feature extraction and a decoder for segmentation mask generation. In this network, the encoder has nine convolutional layers, which are divided into a total of four blocks. Each block consists of two consecutive convolution layers, except for the second block, which consists of three convolution layers. Each convolution layer is followed by a ReLU activation layer. For

spatial reduction of the image, a Max Pooling layer follows at the end of each block. The overall network is a comparatively small network with few parameters, as intended by the authors. To prevent overfitting, blocks two through four have a dropout layer with a dropout rate of 0.2 after each ReLU activation layer. The dropout rate was determined by empirical validation. There are no bottleneck layers between the encoder and decoder in this network. The layer structure of the decoder mirrors that of the encoder. However, instead of convolutional layers, transposed convolutional layers are used, and instead of the Max Pooling layer at the end of the blocks, there is an up-sampling layer at the beginning of the blocks.

We used binary cross-entropy (BCE) as the loss function for training. To optimize the weights during training, the Adam optimizer was used with a learning rate of 0.0001. Training was performed with a batch size of 2 for 100 epochs.

3.3.2. VGG16-UNet

The second neural network for rail track detection is a VGG16-UNet. This network is based on a UNet architecture [31]. This is a widely used and efficient segmentation architecture. This network also uses an encoder and a decoder. The UNet differs from other segmentation networks in that there are skip connections between convolutional layers in the encoder and the corresponding transpose convolutional layers in the decoder. These are used to better incorporate the spatial information of the learned features into the reconstruction of the segmentation map and to obtain a more accurate segmentation result [31]. Due to this feature of the network architecture, we selected this DNN for our experiments. For correct rail track detection, an accurate segmentation is necessary, especially in the more distant parts of the image. In those regions, the rail track area is usually small and hence harder to segment. The UNet architecture is well suited for this task. In this paper, however, we do not use the original UNet architecture of Ronneberger et al., but a modified variant. In the VGG16-UNet, the encoder of the UNet has been replaced by a VGG16 network. This is an efficient architecture for feature extraction that was originally developed for object classification [32]. This network consists of 16 layers, of which 13 are convolutional layers and 3 are fully connected layers. The UNet architecture is retained for the decoder. ReLU activation layers are also used for this network. For the implementation of this network, we used the code of Gupta [33].

To train the VGG16-UNet, also the binary cross-entropy is used as the loss function for training. Stochastic Gradient Descent (SGD) was used as the optimizer for this training, with a learning rate of 0.0001. Training was performed with a batch size of 2 for 100 epochs.

3.3.3. MobileNet–SegNet

As a third neural network for rail track detection, we used a MobileNet–SegNet, which, like the VGG16-UNet, consists of a backbone network for feature extraction, i.e., the encoder, and a decoder network. The MobileNet architecture is used for the encoder part. This was described in a paper by Howard et al. [34] and was developed for computationally efficient image processing in mobile applications. It uses depthwise convolutions instead of conventional convolutional layers. The architecture consists of a convolutional layer followed by 13 depthwise convolutions, each of which is paired with another convolutional layer. Each of these layers is followed by a batch normalization layer and a ReLU activation layer.

The MobileNet–SegNet uses a SegNet architecture [35] as the decoder. Only the decoder part of this architecture is used, which consists of five blocks. Each block starts with an up-sampling layer, followed by a convolution layer and a batch normalization layer. We also used the Gupta implementation for the MobileNet–SegNet [35].

Since the computational resources available for rail track detection in a train are limited, we chose the presented MobileNet architecture for our experiments. This network architecture has been developed specifically for such mobile applications and is therefore well suited for our use case. The SegNet decoder part is intended to support the generation of more accurate segmentation masks. This network has been developed for segmentation tasks in road traffic and is therefore well suited for our use case.

The MobileNet–SegNet uses a similar training configuration to the VGG16-UNet. We trained the MobileNet–SegNet using the SGD optimizer and a learning rate of 0.0001. The loss function is also binary cross-entropy. The batch size is set to 8 for this neural network. In this configuration, training was performed for 100 epochs.

3.4. Confidence Evaluation

A crucial part of the N-version architecture is the confidence evaluation. As can be seen in Figure 1 (architecture diagram), there are two different confidence evaluations throughout the architecture. The first is the NN confidence evaluation, which represents the confidence calculation for each individual rail track detection prediction of the neural networks presented earlier. The other is the confidence combination, which uses the combined prediction and the individual predictions to compute an adjusted overall confidence for the system output.

Both confidence calculation methods are explained in more detail below.

3.4.1. Neural Network Confidence Evaluation

In the Neural Network Confidence Evaluation block of the N-version architecture, the confidence score for the prediction of a neural network is calculated. This confidence score is a form of self-assessment of the respective neural network and fundamentally describes how confident the network is in the respective prediction, i.e., how well the input–output pair could be represented by the learned training knowledge.

In contrast to object recognition in the form of classification, networks for semantic segmentation do not output this confidence score directly. Instead, the segmentation network calculates a probability that each pixel in the image belongs to a certain class. Hence, this output can also be referred to as a probability map.

For the confidence calculation, these pixels of the predicted probability map are assigned to either the Track class or the Background class, depending on their probability value. For this classification, the threshold value θ is used in the equations. This value is later experimentally determined to provide the best possible class division.

With $p_{track}(i)$ as the probability of pixel i belonging to the Track class, N as the total number of pixels in an image and N_{track} and $N_{background}$ as the total number of pixels classified in the respective class, the confidence scores (CS) for the Track and Background classes are calculated by the following equations:

$$CS_{track} = \frac{1}{N_{Track}} \sum_{i=1}^{N} \frac{p_{Track}(i) - \theta}{(1-\theta)} \cdot \mathbb{I}(p_{Track}(i) > \theta) \tag{1}$$

$$CS_{background} = \frac{1}{N_{Background}} \sum_{i=1}^{N} \frac{\theta - p_{Track}(i)}{\theta} \cdot \mathbb{I}(p_{Track}(i) \leq \theta) \tag{2}$$

In this calculation, the indicator function $\mathbb{I}(p_{track,j}(i) > \theta)$ is used to determine whether the probability of a pixel belonging to the Track class exceeds the specified threshold θ. The indicator function returns a value of 1 if the condition is satisfied, and a value of 0 if it is not. Equation (1) calculates the confidence score of the Track class. This score is calculated using the average of the pixel probabilities of all pixels classified into the Track class based on the thresholding. Before calculating the average, every pixel probability is normalized to the

range of probabilities for the Track class (i.e., the range above θ). The higher this value, the more "confident" the neural network is in its prediction of the rail areas in the image.

However, a good prediction of such a segmentation network for rail track detection also requires that it can correctly distinguish the rails from the rest of the image. Therefore, in addition to a high probability for the rail pixels, it is also crucial that the background pixels have a low probability for the Track class. For this reason, Equation (2) calculates the confidence score for the Background class. This calculation also averages the pixel probabilities, which are normalized to the range of probabilities for the Background class. This time, however, this is applied for all pixels whose probability for the Track class is below the threshold θ.

The overall confidence of the neural network's prediction is calculated as the average of the two previously calculated confidence values:

$$CS_{pred} = \frac{CS_{track} + CS_{background}}{2} \tag{3}$$

Only if the network can distinguish between the Background and Track classes with a high probability can it be assumed that the corresponding image is well represented by the training, and the network is therefore confident in its prediction.

With regard to the safety of the overall system, this value can be utilized to detect erroneous or unsafe predictions. The assumption behind this is that the neural networks are not confident in their predictions for situations that do not fit their learned patterns from training. This means that another threshold can be defined to distinguish between predictions that appear uncertain based on the confidence score and those that appear good enough. This confidence threshold is referred to as γ in this paper and can be used to implement a fail-safe behavior for the overall rail track detection system. For predictions with a confidence score below this threshold, the train needs to slow down or come to a stop, in order to implement a safe reaction to the uncertainties in the environment perception. The γ threshold will be determined in a later section.

3.4.2. Confidence Combination

Since the combined prediction of the N-version system is not structurally different from the individual predictions of the neural networks, it is obvious to use the previously presented confidence scores for the combined prediction, resulting in the combination confidence score CS_{comb}. Thus, a rough measure of how confidently the rails were detected by the N-version system can also be found for the combined prediction. However, this confidence score by itself is not significantly more reliable than that of a single neural network, nor does it exploit the strength of such an N-version approach. Due to their structure and the enormous number of trainable parameters, such neural networks have a very high complexity compared to conventional algorithms. The basic idea of the N-version approach is to effectively exploit the complexity of the individual networks. If the predictions of several such very complex systems for complex scenes, such as in the railroad domain, agree, it can be assumed with a high probability that this result is correct.

To exploit this fact, we must find a way to measure the extent to which the individual networks agree on their predictions. The Intersection over Union (IoU) metric is suitable for this purpose. This metric is calculated according to Formula (4), where $Pred$ represents the predicted track pixels and GT represents the track pixels in the ground-truth annotation:

$$IoU = \frac{Pred \cap GT}{Pred \cup GT} \tag{4}$$

In relation to the terms of object detection and classification, this metric can also be defined through the true positives (TP), false positives (FP) and false negatives (FN):

$$IoU = \frac{TP}{TP + FP + FN} \tag{5}$$

For the rest of this paper, the IoU_{GT} represents the Intersection over Union calculated with a prediction and the respective ground-truth annotation. IoU_n will be used for the calculation of the IoU between two neural network predictions.

For the adjusted combined confidence score of the N-version system, the IoUs of all three network predictions are calculated first. Since the IoU calculation is symmetric, three calculations are sufficient in the 3M1I configuration. Finally, the average of these three values is calculated to provide a measure of the agreement of the predictions. As a final step, the previously calculated confidence score for the combined prediction (CS_{comb}) is multiplied by this value. The calculation is shown in Equation (6):

$$CS_{out} = CS_{comb} \times \frac{IoU_1 + IoU_2 + IoU_3}{3} \tag{6}$$

With this combined confidence score (CS_{out}), a more accurate statement can be made about the quality of the overall result of the N-version system. Finally, this score can be used to detect situations where the neural networks do not provide sufficient recognition quality. Such detection allows the implementation of a fail-safe behavior for the operation of the ML-based system. For this purpose, another threshold must be defined to distinguish between sufficiently reliable and insufficient predictions. In the case of insufficient predictions, speed reduction of the autonomous train can be requested in order to allow for more reaction time in situations that are not sufficiently well perceived by the networks. This threshold value was introduced in Section 3.4 as γ and is further discussed in Section 3.6.2.

3.5. Combination Algorithms

For our railway segmentation N-version system, we implemented two different basic combination algorithms. This allows us to gain insights into the significance and potential of such combination algorithms for system reliability. The following section explains both approaches.

3.5.1. Maximum Confidence Voting (MCV)

The main idea for this method is to ensure that the best of the neural network predictions is always forwarded as the system's output. In order to achieve this, the combination algorithm utilizes a form of voting to generate the overall system output. A weight is calculated for each of the predictions, which should describe the quality of the prediction as best as possible. Based on these weights, the algorithm selects the prediction with the highest weight, i.e., the best quality. In our work, we use the Confidence Score (explained in Section 3.4) as a simple way of assessing this quality. Hence, the MCV algorithm always selects the individual neural network prediction with the highest confidence as the combined output.

3.5.2. Pixel Majority Voting (PMV)

As a second combination algorithm, we implemented a different form of voting. In contrast to the first algorithm, this method, which will be called Pixel Majority Voting (PMV), is not about selecting the best of the three predictions. Instead, the goal is to achieve the best possible combination of all three predictions. For this purpose, voting is also performed in this method, but this time at the pixel level. Since there are three predictions for the same scene, majority voting can be implemented. This means that for each pixel in

all three predictions, it is evaluated whether this pixel belongs to the Track or Background class. With $p_{Track,j}(i)$ as the probability of pixel i in prediction j belonging to the Track class and θ as the probability threshold for classification as part of the Track (>θ) or Background ($\leq \theta$) class, the voting is implemented by the following equation:

$$class(i) = \begin{cases} Track, & \text{if } \sum_{j=1}^{3} \mathbb{I}(p_{Track,j}(i) > \theta) \geq 2 \\ Background, & \text{else} \end{cases} \tag{7}$$

Based on the indicator function, pixel i in the combined prediction is assigned the respective $class(i)$ predicted by at least two of the networks.

Since the combined prediction should be similar to the individual network predictions, in order to allow for the same confidence calculation methods, only classifying a pixel for the combined prediction is not sufficient. In order to achieve this, a combined probability value for every pixel needs to be calculated. This is simply achieved by calculating the average probability for all predictions that voted for the selected ("winner") class. Equations (8) and (9) show these calculations for the respective classes.

$$p_{Track,comb}(i) = \frac{1}{\sum_{j=1}^{3} \mathbb{I}(p_{Track,j}(i) > \theta)} \sum_{j=1}^{3} p_{Track,j}(i) \cdot \mathbb{I}(p_{Track,j}(i) > \theta) \tag{8}$$

$$p_{Background,comb}(i) = \frac{1}{\sum_{j=1}^{3} \mathbb{I}(p_{Track,j}(i) \leq \theta)} \sum_{j=1}^{3} p_{Track,j}(i) \cdot \mathbb{I}(p_{Track,j}(i) \leq \theta) \tag{9}$$

3.6. Threshold Optimization

The previously described N-version architecture for railway detection utilizes two threshold values. The relation between both thresholds and the introduced terminology is briefly described in Figure 2. Both values need to be optimized for optimal system performance and failure detection. The optimization process for each of the threshold values is described in the following subsections.

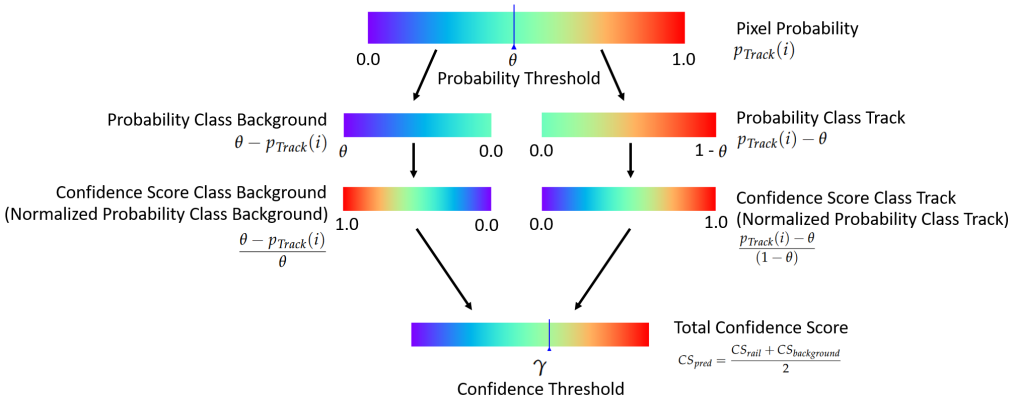


Figure 2. Definition and relation of θ and γ thresholds with respect to pixel probabilities of the rail track detection prediction.

3.6.1. Theta (θ)

The θ threshold is an important property of the implemented N-version architecture. It is the basis for the correct classification of the classes in the predictions as well as for the calculation of the confidence. The goal of this section is to determine an optimal value for this threshold for the N-version architecture presented in the previous sections.

An obvious assumption for the θ threshold value would be a limit of 0.5, i.e., 50% confidence. Nevertheless, an investigation was first carried out in this paper to determine the actual optimal threshold value. For this purpose, thresholds between 0 and 1 (i.e., 0% to 100%) were iterated in steps of 0.01 for the used test set (2,328 images), and the average IoU for the predictions of each network was calculated based on the threshold. The result of this analysis is shown graphically in Figure 3 (θ optimization graph).

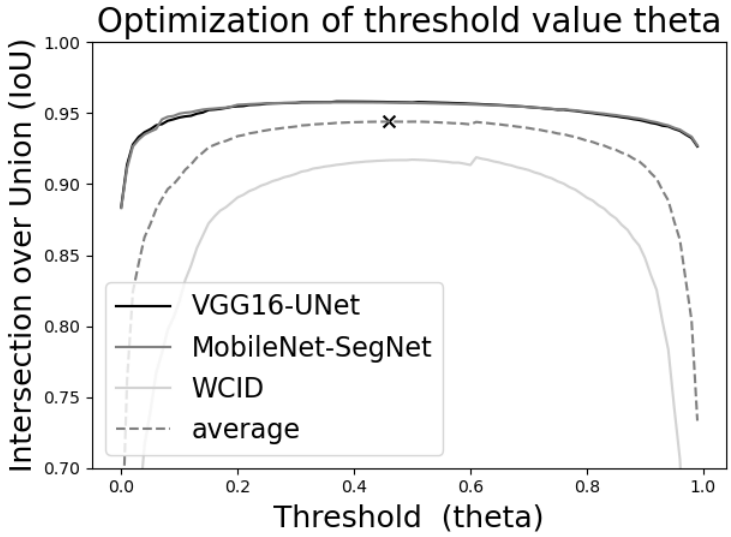


Figure 3. Optimization of threshold θ to distinguish between Track and Background classes.

In this analysis, it can be seen that the graphs of the three neural networks (shown by the solid lines) have a very similar shape. In the middle value range of θ thresholds (between 0.2 and 0.8), there is a flat curve with high IoU values. On the other hand, at the edges, there is a significant drop in the average IoU values. It is also noticeable that the curves for the VGG16-UNet (black) and the MobileNet–SegNet (gray) are almost identical with very high IoU values of about 0.95. Even though the WCID network shows a similar curvature at the edges of the graph, the middle part of the curve lies below the other two and shows – different than the other networks – no distinctive plateau in the center of the graph. Therefore, it can be concluded that the WCID network provides slightly worse detection results than the VGG16-UNet and the MobileNet–SegNet, but overall, all three networks show good detection performance. The drop in the IoU curves for very high and very low θ thresholds also shows that the neural networks have learned a robust distinction between the Track and Background classes through training. Most pixels seem to have either a very high or very low probability for the Track class in the predictions, because only in this range does a small change in the threshold cause a large change in the IoU values, i.e., in the correct classification of the pixels.

Because of the large flat areas in the middle of the graph, it is difficult to determine an optimal θ value that will produce the highest average IoU values. In order to determine an optimal threshold, the average of the three neural networks is calculated and shown as the dashed line in the figure. The maximum of this curve is at a threshold of 0.46. At this point, all three individual networks together provide the best predictions. Therefore, for analyses in this paper, a θ threshold of 0.46 is used to distinguish between the Track and Background classes.

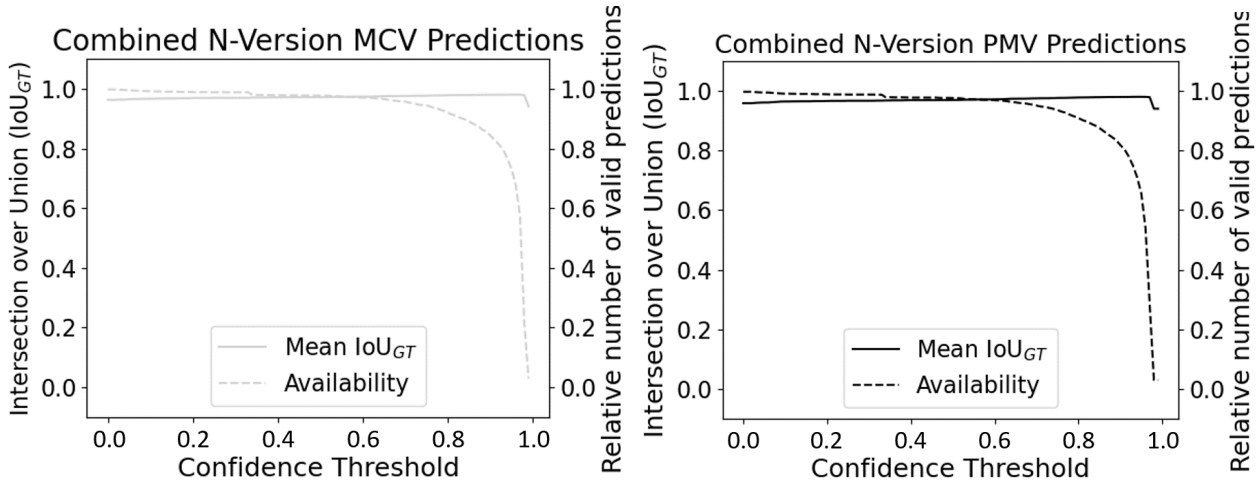
3.6.2. Gamma (γ)

As introduced earlier, the detection of the train's path is a safety-critical function in an autonomous train. With the computed confidence of the combined prediction, the confidence of the N-version architecture in each of its predictions can be analyzed. This

confidence value is the only way to draw conclusions about the quality of the predictions while the train is moving, without the availability of ground-truth data to verify the predictions. In order to effectively identify unsafe predictions during operation with the N-version architecture presented earlier, it is therefore necessary to distinguish between supposedly safe and unsafe predictions based on this combined confidence value. The γ threshold is used for this purpose. The γ threshold is hence an essential parameter of the N-version architecture. The goal of this section is to find the best possible value for this threshold.

As a parameter of the N-version architecture, the γ threshold has a direct impact on two system attributes that are directly related to the dependability of the system. First, by detecting and filtering out low-confidence predictions based on this threshold, the safety of the predictions, and thus of the system itself, can potentially be increased. This means that the predictions that are above the threshold and are therefore marked as valid have a higher probability of not containing safety-critical errors. Thus, the safety of the system is supposed to be positively influenced. On the other hand, the second system property, availability, is reduced by filtering out unsafe predictions. The lower the confidence threshold is set, the more predictions are classified as invalid and the system cannot perform its specified function in these situations. The higher the rate of excluded predictions, the lower the overall availability of the system.

Therefore, when optimizing the γ threshold as a system parameter, it is necessary to achieve the best possible trade-off between increasing the safety of the predictions and reducing the availability of the system. Such an investigation has been carried out for the presented N-version architecture. The influence of the confidence threshold was analyzed separately for both combination algorithms (i.e., MCV and PMV). The results are shown in Figure 4 (γ optimization graphs).



(**a**) (**b**)

Figure 4. Optimization of confidence threshold γ according to safety vs. availability trade-off for (**a**) Max Confidence Voting (MCV) and (**b**) Pixel Majority Voting (PMV).

In the two graphs, confidence thresholds in the range of 0.0 to 1.0 were examined separately for each combination algorithm (MCV on the left and PMV on the right) in increments of 0.01 (i.e., the x-axis). Since the graph shows both the safety and the availability of the system, two y-axes are shown. The average IoU_{GT} of all predictions in the test set whose confidence is above the respective threshold is shown on the left y-axis and with the solid curve. This metric describes the quality of the predictions, but does not allow direct conclusions about safety-critical errors in the predictions. A suitable metric to describe such safety-critical errors is, for example, the Track Center Offset Metric (TCOM) [36]. For this

metric, however, the distance of the predicted ego-track to the ground-truth ego-track must be calculated. For single RGB images, such distance calculations are not easily possible. Therefore, the analysis performed is based on the assumption that if the quality of the predictions is high (i.e., a high IoU_{GT}), there will be fewer safety-critical errors in the predictions. As a measure of availability, we use the ratio of the number of images filtered out using the respective threshold, to the total number of images in the test set used. These values are shown on the right y-axis and with the dashed curve.

The two graphs are almost identical. It can be seen that, as expected, more and more predictions are filtered out as incorrect as the confidence threshold increases. Especially for high thresholds above 0.8, the availability drops rapidly to 0.0. A large part of the predictions seem to be in this confidence range. It can also be seen that the γ threshold has a positive effect on the quality of the predictions. The higher the threshold value, the more predictions are classified as invalid, which also increases the quality, and thus presumably the safety. This correlation is almost linear for both combination algorithms, with a positive increase. However, this increase is very small for both algorithms. To find the optimal value for the confidence threshold, the intersection of the two curves is determined. At this point, the safety and the availability are at the same time the highest, which leads to the best system performance. This intersection is at 0.56 for both MCV and PMV. This value for γ is used in the Section 4.

4. Results

This section presents the results from the analysis of the previously described N-version architecture for railroad segmentation. All parts of this architecture were analyzed individually using the dataset described in Section 3.2.

4.1. Neural Network Training Results

As the curves for the optimization of the θ threshold in Figure 3 (θ optimization graph) in Section 3.6.1 indicate, all three trained neural networks show very good perception performance on the test set used. However, to examine the results in more detail, histograms of the IoU of the predictions for each neural network were generated. These can be seen in Figure 5 (individual network IoU histograms).

The histograms confirm the good recognition performance and robustness of the trained networks. They also show the very similar quality of the predictions of VGG16-UNet and MobileNet–SegNet and the comparatively worse performance of the WCID network. In the case of VGG16-UNet and MobileNet–SegNet, the predictions for more than half of the 2328 test images have an IoU above 0.98, which is the highest bar in the histogram. For the WCID network, most of the predictions are in the range 0.96–0.98, with slightly fewer in the highest range. This also shows that WCID predicts the rails slightly worse than the other two networks. It is also noticeable in all three histograms that there are hardly any predictions in the range below 0.8. This means that the networks generally provide very good predictions. Only in the range 0–0.02 are there some predictions for each network that can be characterized as error cases due to the very low agreement with the ground truth. Figure 6 (network predictions) shows a positive and a negative example from the qualitative analysis of the railroad predictions. The images are overlaid with a heatmap visualization of the pixel probabilities for the Track class. In the top left corner, the respective IoU and confidence value for the particular prediction are shown in white.

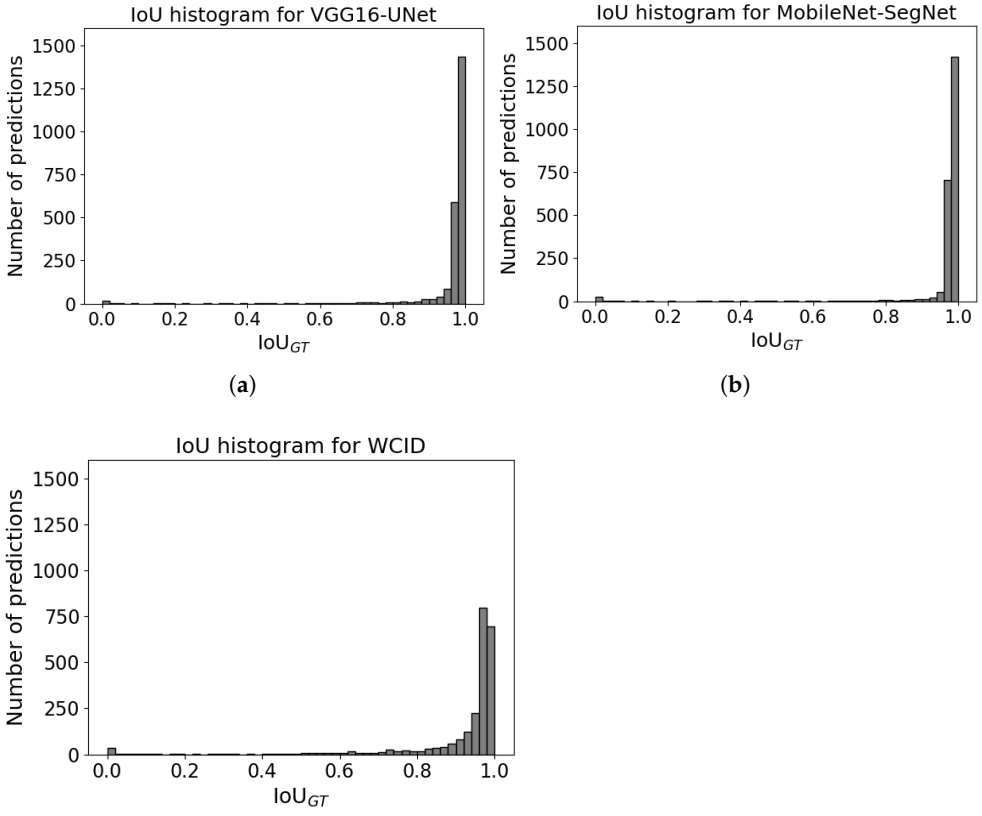


Figure 5. IoU histogram analysis for (**a**) VGG16-UNet network (**b**) MobileNet–SegNet network and (**c**) WCID network.

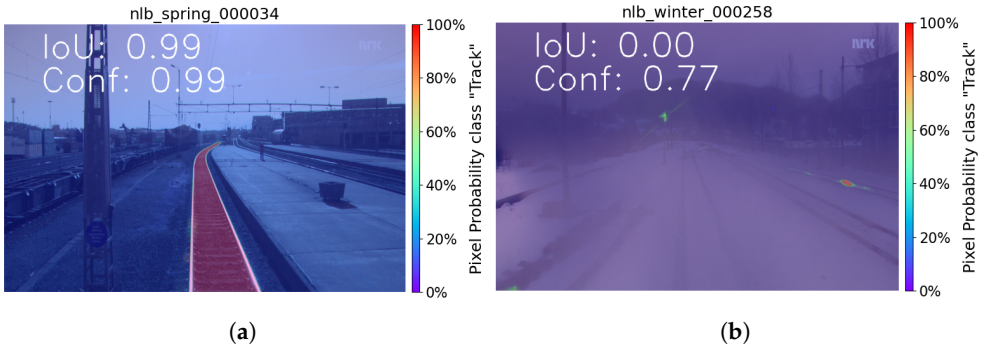
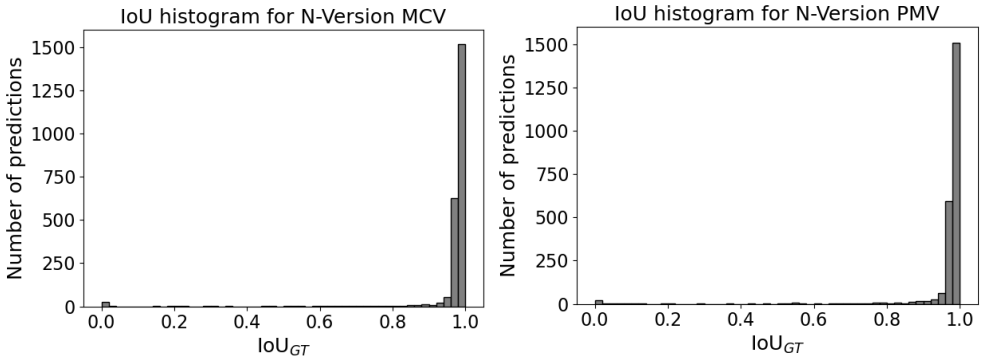


Figure 6. Example predictions for the neural networks on the test set. The images show (**a**) a good prediction from the VGG16-UNet network and (**b**) a bad prediction from the MobileNet–SegNet network.

4.2. Combination Algorithm Results

In this section, the results of the two combination algorithms defined in Section 3.5 are presented. Overall, it can be said that both combination algorithms show very good performance in recognizing the rail areas. This can also be seen for the combination algorithms using the IoU histograms (shown in Figure 7). Both algorithms show a similarly good detection performance as the VGG16-UNet and the MobileNet–SegNet. Most predictions are in an IoU range of above 0.98. Almost no predictions have an IoU below 0.8. This means that the combination algorithms were able to detect the rails for the majority of the test images and to predict their position in the image almost correctly. From the two histograms, it can be seen that the PMV combination algorithm has a slightly better overall quality in

its predictions, as there are more predictions in the range of an IoU above 0.98 and fewer in the range between 0.96 and 0.98. In the following, the results of the combination algorithms are analyzed in more detail and compared with those of the individual networks.



(**a**) (**b**)

Figure 7. IoU histogram analysis for (**a**) Max Confidence Voting (MCV) combination algorithm and (**b**) Pixel Majority Voting (PMV) combination algorithm.

4.2.1. Maximum Confidence Voting (MCV)

Since the prediction quality (i.e., IoU) of the MCV combination algorithm cannot, by definition, be higher than that of the best individual network, this combination algorithm does not allow for any improvement in prediction quality. However, it is also possible that the best prediction of the three individual networks is not selected using the MCV method. In this case, the overall prediction quality of the N-version system is reduced compared to the individual networks. Therefore, an analysis was performed to investigate the ability to select the best prediction using the MCV algorithm, i.e., based on the confidence of the individual predictions. The results are shown in Table 3.

Table 3. N-version MCV results on NLB test set (2,328 images).

θ Threshold	Best Prediction	Middle Prediction	Worst Prediction	VGG16-UNet	MobileNet–SegNet	WCID
0.46	1,325 (56.9%)	762 (32.7%)	241 (10.4%)	1,341 (57.6%)	982 (42.2%)	5 (0.2%)

The table shows that the MCV combination algorithm is able to select the prediction with the highest IoU, i.e., the best quality, from the three individual predictions for more than half (56.9%) of the test images. Furthermore, the MCV algorithm selected the prediction with the middle of the three IoU values for approximately one third of the test images, and the individual prediction with the worst IoU for just over 10% of the images. Overall, it can be said that selection based only on the confidence of the individual predictions usually works well. Figure 8 (MCV worst-case selection) shows an example of such errors. This incorrect selection usually occurs when at least one of the networks is very overconfident in its prediction, which is what can be seen in the example image. The top row and the bottom left image depict the predictions of the three individual neural networks for the rail area. As can be seen, the MobileNet–SegNet (i.e., top right) does not detect any rail areas in this image. Since this scene is in a tunnel, the tracks are very hard to see due to the poor lighting. Nevertheless, both the VGG16-UNet and the WCID network are able to detect at least parts of the track correctly and thus have an IoU_{GT} greater than 0. However, it can also be seen that the MobileNet–SegNet with a confidence of 1.0 is very confident in its own prediction, even if it is wrong.

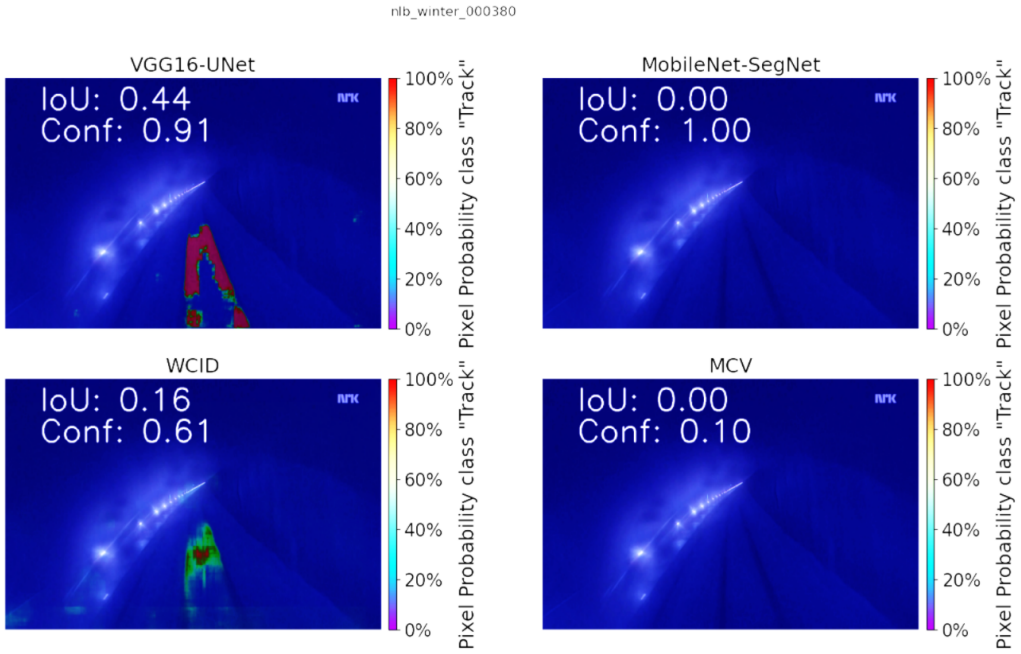


Figure 8. Example of a selection of the worst individual prediction by the Max Confidence Score (MCV) combination algorithm. The image shows the predictions of all three individual neural networks (**top row** and **bottom left**), as well as the selected output from the MCV algorithm (**bottom right**). The respective IoU and confidence values for every prediction are shown in white in the top left corner of every prediction.

Overall, such incidents do only occur rarely. They were observed for the following number of test set images for every network: MobileNet–SegNet (10); VGG16-UNet (5); WCID (10). When using the MCV combination algorithm, such constellations can lead to bad predictions. In the whole test set, the image shown in Figure 8 (MCV worst-case selection) is the only instance where such erroneous predictions with maximal confidence were selected as the overall output using the MCV combination algorithm.

The table also shows that the MCV algorithm selects the VGG16-UNet predictions most frequently (57.6% of the time). The MobileNet–SegNet predictions are also very often included in the overall prediction of the MCV algorithm (42.2%). At the same time, this means that the WCID predictions are almost never selected for the overall prediction of the MCV algorithm. This observation is consistent with the results from Section 4.1, indicating that WCID provides the worst predictions of the three networks and the other two have a similarly high performance. In addition, this result also shows that the training of the WCID still worked well. Although the network provides predictions whose quality does not match that of the other networks, the WCID also provides a confidence level that matches this lower quality, thus allowing the selection of better-quality predictions. This behavior is very important for the MCV combination algorithm. Although the individual networks may provide less good predictions, they must not be overconfident. This seems to be the case for the three trained networks, which means that the MCV algorithm performs quite well overall.

4.2.2. Pixel Majority Voting (PMV)

In contrast to MCV, the PMV combination algorithm is able to produce better combined predictions compared to the individual network predictions. A comparison with the individual predictions is also very interesting for this algorithm, but other comparison categories are necessary. To this extent, we examined how often the predictions combined with PMV each have a higher IoU than the best, middle, and worst individual predictions.

Also for this combination algorithm, it is possible to produce predictions inferior to the predictions of the individual networks. The frequency of this case was also evaluated in the analysis. The results of this evaluation can be seen in Table 4.

Table 4. N-version PMV results on NLB test set (2,328 images).

θ Threshold	IoU > Best Prediction	IoU > Middle Prediction	IoU > Worst Prediction	IoU < Worst Prediction
0.46	746 (32.0%)	1,212 (52.1%)	357 (15.3%)	13 (0.6%)

As this evaluation shows, the PMV combination algorithm is able to deliver very good results. If we compare this algorithm with the individual predictions (as shown in Table 4), it becomes clear that this algorithm, using pixel-by-pixel voting, leads to a prediction with a higher IoU than the best individual network for 32% of the test images. Overall, almost 85% of the predictions also have a higher IoU than the middle individual network. Furthermore, there are only 13 images in the entire test dataset for which the PMV algorithm predicts worse than the worst single network. In all those 13 images, the difference in the IoU_{GT} between the PMV prediction and the worst individual prediction never exceeded 0.01. This means that the quality of these inferior predictions of the PMV combination algorithm is still almost at the level of the worst individual prediction.

Of particular interest are the cases where the combined result is of higher quality than the individual predictions. In this case, pixel-by-pixel voting seems to be able to improve the individual predictions and correct errors. To examine this in more detail, Figure 9 (PMV error correction) shows an example of such error corrections.

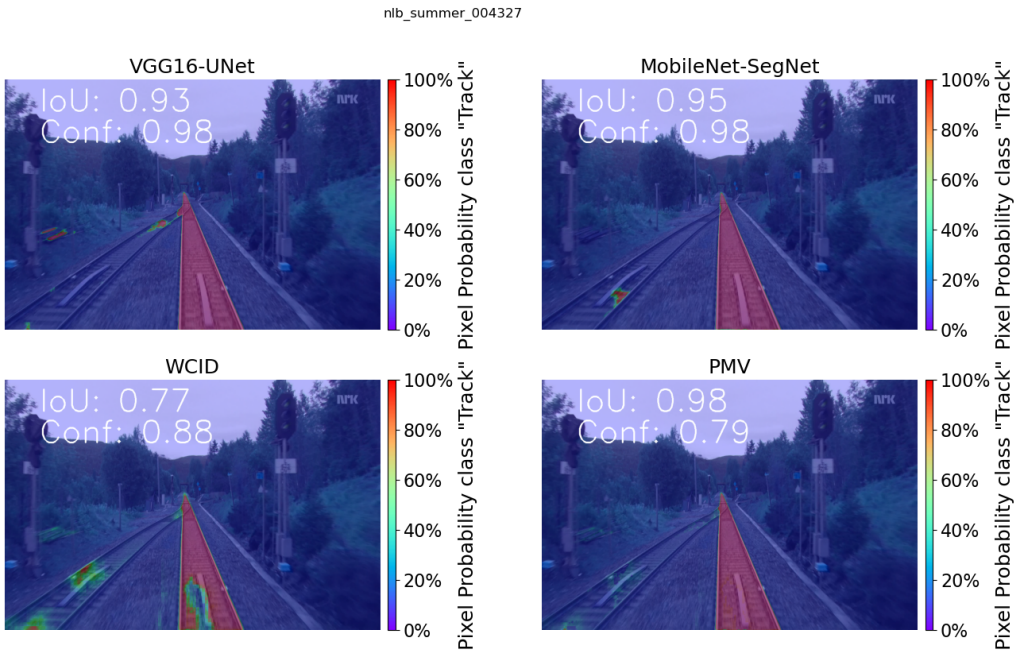


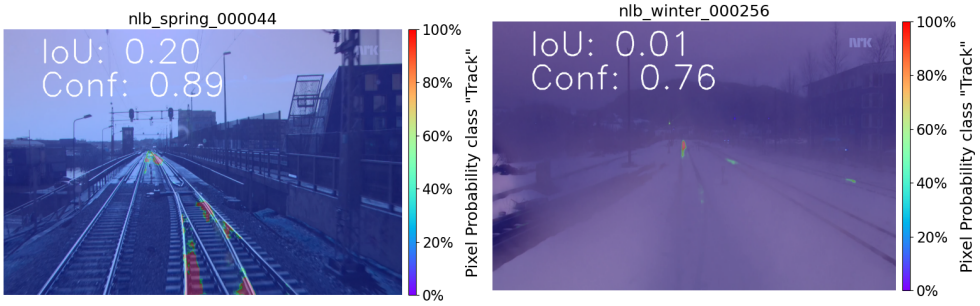
Figure 9. Example prediction for the Pixel Majority Voting (PMV) combination algorithm with higher quality than all individual networks. The image shows the predictions of all three individual neural networks (**top row** and **bottom left**), as well as the selected output from the MCV algorithm (**bottom right**). The respective IoU and confidence values for every prediction are shown in white in the top left corner of every prediction.

In this example, it can be seen that all individual neural networks manage to recognize the ego-track quite well. Only the WCID prediction shows a gap in the ego-track area. Additionally, all the individual networks predict an additional rail area on the left side

of the ego-track. Due to the fact that every neural network produces a different error in its prediction, the PMV combination algorithm is able to correct these errors and hence predicts only the correct ego-track area. The improvement in the prediction can also be seen in the respective IoU_{GT} values of the predictions.

4.3. Confidence Results

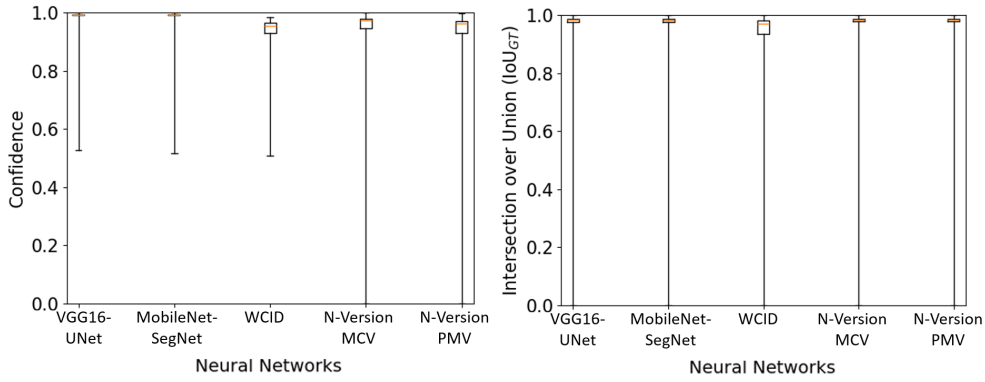
The previous analyses only examined the quality of the predictions in the form of the IoU_{GT}. However, in addition to this prediction quality, the confidence value of the prediction also plays an important role in safety-critical systems. This value indicates how confident the network or system is in the quality of its prediction. Since neural networks are stochastic systems, their predictions are only estimates and therefore subject to error. In a safety-critical system, such as rail track detection for trains, it is therefore necessary to be able to detect these errors in the best possible way and handle them safely. The confidence value can be used to identify these errors during operation of the system. If the neural network's confidence in its prediction is below a certain threshold, it can be assumed that the prediction is incorrect and the train can be slowed down or stopped to give the system more time to correctly assess the situation. To do this, the confidence value must correlate well with the actual prediction quality. However, as [37] found in their work, neural networks are very often overconfident in their predictions after training. This means that the confidence value is higher than the actual quality of the predictions. Also, our analyses showed such behavior for the individual neural networks. The confidence value for the predictions of the neural networks for rail track detection was calculated according to the formula in Section 3.4. Figure 10 (example predictions) shows some example images from the test set for which the individual networks provide too-high confidence values for their prediction, thus preventing the detection of errors.



(**a**) (**b**)

Figure 10. Example predictions for the individual neural networks with a confidence much higher than the IoU_{GT}. The images show (**a**) a VGG16-UNet network prediction and (**b**) a MobileNet–SegNet network prediction. The images are overlaid with a heatmap visualization of the respective network prediction. In the top left corner, the IoU_{GT} and confidence value for every prediction are shown in white.

To examine this behavior statistically over the entire test set, the distribution of the confidence values for each of the individual networks is shown in a box plot in Figure 11 (box plot analysis) on the left side and compared with the distribution of the IoU_{GT} values, i.e., the prediction quality, on the right side. Five box plots are shown in each of the two graphs, representing both the three individual networks (on the left) and the results of the two combination algorithms of the N-version system (on the right).



(**a**) (**b**)

Figure 11. Box plot analysis for (**a**) the confidence distribution and (**b**) the IoU_{GT} distribution. In both boxplots, the individual neural networks are represented by the three bars on the left and the N-version combination algorithms are represented by the two bars on the right respectively.

Comparing the confidence values with the prediction quality, it is noticeable that the IoU_{GT} values for the individual neural networks are distributed over the entire range from 0.0 to 1.0, with the majority of the predictions of each network lying in a very narrow range above 0.9. Most of the confidence values are in a very narrow and high range too, but it is also noticeable that the boxes here are higher and smaller than the ones for the corresponding IoU_{GT} values. In addition, the confidence values are only distributed in the range above about 0.5. These two aspects indicate that the confidence of the predictions of the individual networks is a poor reflection of their actual quality. This means that these networks individually would be poorly suited for use in a safety-critical system. In contrast, the figure shows that for the combined results of the N-version system for both combination algorithms used, the confidence distribution is significantly closer to the IoU_{GT} distribution. It can also be seen that the boxes in the confidence distribution for both algorithms are significantly larger than the corresponding boxes in the IoU_{GT} distribution. This means that the combination algorithms estimate their own confidence somewhat more conservatively. Another indication for this is the observation that the confidence distribution covers the entire range from 0.0 to 1.0.

In addition to the statistical distribution of the confidence, the correctness of the confidence was also analyzed. As Equations (1) and (2) in Section 3.4 show, the confidence score is mainly calculated from the individual pixel probabilities of an image. This means, in order to analyze the correctness of the confidence score, we can also analyze the correctness of the pixel probabilities. Consequently, we analyzed, for each pixel probability in the Track class, the frequency with which the corresponding pixels were classified as Track in the respective ground-truth mask. This allows us to determine how well the networks can estimate their own uncertainty. The following figure shows the corresponding graphs for this analysis for each of the individual networks (solid lines) and for the results of the combination algorithms (dashed lines).

In Figure 12, the optimal curve is shown in red as a straight line with a slope of 1.0. On this optimal curve, every predicted pixel probability for the Track class (i.e., the confidence on the x-axis) exactly matches the relative number of the same pixels classified as Track in the respective ground-truth images on average over the test set. This would mean, for example, that for all pixels predicted with a probability of 0.5 for the Track class, exactly 50 % of the respective pixels in the ground-truth masks were also labeled in the Track class.

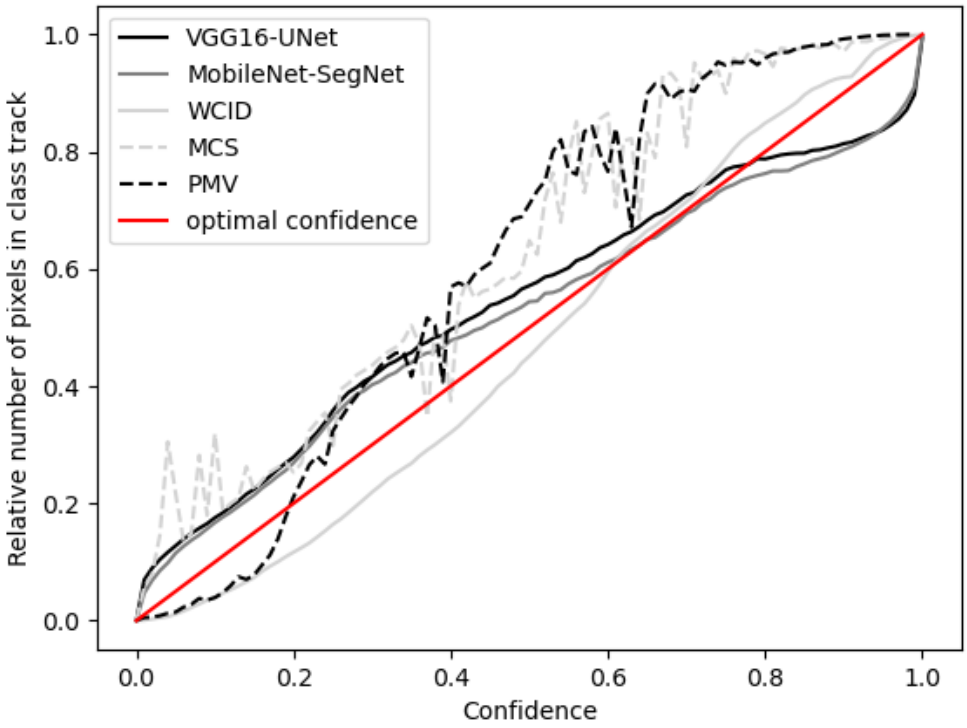


Figure 12. Analysis of the correctness of the confidence values.

All values above this line represent a conservative estimate of their own confidence, where the pixels are classified as Track more often in the ground truth than the confidence indicates. Values below the red line indicate an overconfident estimate of confidence. The graph shows that all of the individual networks (shown as solid lines) lie closely around this optimal straight line. All three networks also show areas of too much confidence (below the red line) and too little confidence (above the red line). As in the other analyses, the MobileNet–SegNet and the VGG16-UNet show a very similar trend. For all confidence thresholds below 0.6, both show conservative behavior. Here, they estimate their own confidence worse than the ground truth. For higher confidence thresholds, however, both networks overestimate themselves very much. Here, the confidence of these networks cannot be used well to find errors in their predictions. Compared to these two networks, the WCID network shows a curve that appears almost mirrored on the optimal straight line (i.e., the red straight line). This network overestimates its prediction quality in the lower threshold range and underestimates it in the upper threshold range. Again, the intersection with the red line is in the range of 0.6–0.7. Overall, therefore, none of the three individual networks is consistently reliable in assessing its own prediction performance over the full range of confidence levels. The situation is different for the combined N-version results. Both networks' confidence estimates are in the conservative range above the red line for almost all confidence values. This means that they almost always estimate their confidence value to be somewhat lower than their actual prediction performance.

5. Discussion

In this work, an N-version architecture for rail track detection using neural networks was investigated. The results show that the three networks used achieved different but overall good training performances. In particular, the WCID network, which was deliberately chosen as a small model, showed a lower performance compared to the other two architectures, as expected. Nevertheless, all models achieved very good results in terms of rail track detection, measured against the IoU_{GT}.

Despite the high overall detection performance, it was also possible to identify some test images where the quality of the predictions was insufficient. This underscores the weaknesses of neural networks discussed in Section 2, especially in safety-critical applications. It was also noticeable that high confidence values often occurred even with inaccurate predictions. This shows that the confidence values of individual networks alone are not sufficient to reliably assess the safety of the segmentation results.

As the experimental results show, the used N-version architecture helps to improve the safety of the predictions. While the MCV algorithm reliably selected the prediction of the best single network in the majority of cases, the PMV algorithm achieved a higher prediction quality than the best single network for many images. A detailed analysis of the test results also revealed several cases where the PMV algorithm successfully corrected the errors of the individual DNNs. This was demonstrated in Section 4.2 using a test image as an example. This showed that both the MCV and PMV algorithms are suitable methods for combining the individual networks in the N-version architecture.

In contrast to related N-version approaches from the literature [23–25], in our work, we implemented a complete concrete N-version approach for a real and safety-critical use case. Furthermore, we implemented two different combination algorithms and were able to investigate the influence of such algorithms. In addition, we developed a confidence computation method adapted to the use case, which can be used as a safety indicator at runtime.

Applying N-version architectures in ML-enabled operational rail systems is a promising path towards certification of these systems. Current railway safety standards such as EN50716:2023 [20] acknowledge the importance of machine learning technology for future railway applications, but also highlight the challenges associated with this technology and preconditions for their use in safety-critical contexts. As shown in safety-critical applications in other sectors, N-version architectures can help to address some of these points, similar to the application of N-version architectures when implementing traditional software-enabled systems satisfying very high safety integrity levels [38]. To pave the way towards certifying ML-enabled N-version systems, further work is required to achieve and demonstrate basic safety integrity levels for the individual neural networks, and to better understand the behavioral diversity of the networks that is key for fault detection and fault masking.

Despite the promising results, there are some limitations that should be addressed in future work. A key point is the composition of the test dataset, which mainly contains images of low complexity. To further improve the validity of the results, future studies should include more complex scenarios.

So far, the diversity of the implemented N-version architecture is based solely on model diversity. The results show that this measure can already lead to first improvements regarding the prediction safety. However, to realize the full potential of the approach, other sources of diversity should be included. These include training on different datasets to ensure greater generalization capability, using different cameras or sensor technologies to achieve robustness to different environmental conditions, testing additional combination algorithms that may enable better prediction fusions, as well as using alternative methods for confidence calculation and plausibility checks to better quantify the quality and safety of the predictions.

Furthermore, it can be seen that the Intersection over Union (IoU) is not sufficient as a sole evaluation metric for rail track detection. Especially in real-world applications, a more accurate evaluation of the centerline error is important. Only such metrics allow the evaluation of safety-critical failures in the predictions in contrast to only a prediction

quality evaluation using the IoU_{GT}. Besides that, statistical significance testing for the IoU improvements is also important and needs to be performed in the future.

For use in an operational environment, it would also be necessary to pay attention to the real-time performance of the ML-enabled system, especially in light of the typically limited hardware resources of on-board computing systems. Whilst the adaptation of ML network architectures and algorithms to limited hardware resources may lead to degraded inference performance, N-version architectures can help to counterbalance these drawbacks by combining the results of the real-time-capable networks. Assuming that these challenges are met, the approach can be transferred to other domains with similar technical challenges and certification environments, such as aerospace, automotive, and industrial robotics.

When developing an ML-based N-version architecture, the increased effort for selecting and training the neural networks must also be taken into account. Compared to a single network, the time and computational resources required increase with each additional network in the N-version architecture. Furthermore, the achieved model diversity is strongly dependent on the type of training data and the selected model architectures. It must be further examined how stable the advantages of the approach are under other conditions or on other datasets.

Our results raise several interesting topics for future work. A more detailed investigation of the influence of specific diversity criteria in the N-version architecture may provide further clarity on the advantages of this approach. An integration of explainable ML components that make the differences between the models interpretable also seems promising. Finally, an investigation of how N-version learning can be integrated into formal safety cases is also necessary in the future.

6. Conclusions

In summary, this work shows that the N-version architecture offers a promising approach to improve the safety of rail track detection using neural networks. In particular, by using the presented methods for confidence combination, erroneous predictions can be detected more reliably at runtime and, as shown, can even be partially corrected by selecting an appropriate combination algorithm for the predictions. At the same time, there are still several open research questions, especially regarding the diversity of the models, the plausibility of the predictions, and the choice of suitable evaluation metrics. Further investigation of these aspects could help to further optimize the approach and increase its practical suitability for safety-critical applications.

Author Contributions: Conceptualization, P.J.; methodology, P.J.; software, P.J.; validation, P.J. and C.T.; formal analysis, C.T.; investigation, P.J.; resources, P.J.; data curation, P.J.; writing—original draft preparation, P.J.; writing—review and editing, C.T.; visualization, P.J.; supervision, C.T.; project administration, C.T.; funding acquisition, C.T. and P.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research is supported by the CertML project (2022–2025). The project 'CertML—Certifiable machine learning based controls for safety-critical applications' is funded by the German Federal Ministry of Education and Research (BMBF) as part of the program 'KI4KMU—Research, Development and Use of Artificial Intelligence Methods in SMEs' (Grant Number 01IS22029C). The APC was funded by CertML as well.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: This work was supported by HTW Berlin student Biranavan Parameswaran. He provided coding and training of neural networks.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

CNN	Convolutional neural network
DMSI	Double-model single-input
DNN	Deep Neural Network
GoA	Grade of Automation
IoU	Intersection over Union
MCV	Maximum Confidence Voting
MDPI	Multidisciplinary Digital Publishing Institute
ML	Machine learning
PMV	Pixel Majority Voting
TMTI	Triple-model triple-input

References

1. Grigorescu, S.; Trasnea, B.; Cocias, T.; Macesanu, G. A survey of deep learning techniques for autonomous driving. *J. Field Robot.* **2020**, *37*, 362–386. [CrossRef]
2. Amodei, D.; Olah, C.; Steinhardt, J.; Christiano, P.; Schulman, J.; Mané, D. Concrete problems in AI safety. *arXiv* **2016**, arXiv:1606.06565.
3. Zhao, X.; Banks, A.; Sharp, J.; Robu, V.; Flynn, D.; Fisher, M.; Huang, X. A safety framework for critical systems utilising deep neural networks. In Proceedings of the International Conference on Computer Safety, Reliability, and Security, Lisbon, Portugal, 15–18 September 2020 ; Springer: Berlin/Heidelberg, Germany, 2020; pp. 244–259.
4. Dmitriev, K.; Schumann, J.; Holzapfel, F. Toward Certification of Machine-Learning Systems for Low Criticality Airborne Applications. In Proceedings of the 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA, 3–7 October 2021; pp. 1–7. [CrossRef]
5. *DIN EN 62290-1: 2014*; Bahnanwendungen–Betriebsleit-und Zugsicherungssysteme für den städtischen schienengebundenen Personennahverkehr—Teil 1: Systemgrundsätze und Grundlegende Konzepte. Beuth: Berlin, Germany, 2015 .
6. Singh, P.; Dulebenets, M.A.; Pasha, J.; Gonzalez, E.D.R.S.; Lau, Y.Y.; Kampmann, R. Deployment of Autonomous Trains in Rail Transportation: Current Trends and Existing Challenges. *IEEE Access* **2021**, *9*, 91427–91461. [CrossRef]
7. Aoun, J.; Quaglietta, E.; Goverde, R.M.P. Investigating Market Potentials and Operational Scenarios of Virtual Coupling Railway Signaling. *Transp. Res. Rec. J. Transp. Res. Board* **2020**, *2674*, 799–812. [CrossRef]
8. Trentesaux, D.; Dahyot, R.; Ouedraogo, A.; Arenas, D.; Lefebvre, S.; Schon, W.; Lussier, B.; Cheritel, H. The Autonomous Train. In Proceedings of the 2018 13th Annual Conference on System of Systems Engineering (SoSE), Paris, France, 19–22 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 514–520. [CrossRef]
9. Hyde, P.; Ulianov, C.; Liu, J.; Banic, M.; Simonovic, M.; Ristic-Durrant, D. Use cases for obstacle detection and track intrusion detection systems in the context of new generation of railway traffic management systems. *Proc. Inst. Mech. Eng. Part F J. Rail Rapid Transit* **2022**, *236*, 149–158. [CrossRef]
10. Li, H.; Zhang, Q.; Zhao, D.; Chen, Y. RailNet: An Information Aggregation Network for Rail Track Segmentation. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–7. [CrossRef]
11. Tao, Z.; Ren, S.; Shi, Y.; Wang, X.; Wang, W. Accurate and Lightweight RailNet for Real-Time Rail Line Detection. *Electronics* **2021**, *10*, 2038. [CrossRef]
12. Belyaev, S.; Popov, I.; Shubnikov, V.; Popov, P.; Boltenkova, E.; Savchuk, D. Railroad semantic segmentation on high-resolution images. In Proceedings of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), Rhodes, Greece, 20–23 September 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1–6. [CrossRef]
13. Ristić-Durrant, D.; Franke, M.; Michels, K. A Review of Vision-Based On-Board Obstacle Detection and Distance Estimation in Railways. *Sensors* **2021**, *21*, 3452. [CrossRef] [PubMed]
14. Yang, S.; Yu, G.; Wang, Z.; Zhou, B.; Chen, P.; Zhang, Q. A Topology Guided Method for Rail-Track Detection. *IEEE Trans. Veh. Technol.* **2022**, *71*, 1426–1438. [CrossRef]

15. Yang, S.; Wang, Z.; Yu, G.; Liu, W. Key Point Estimate Network for Rail-Track Detection. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 4077–4088. [CrossRef]
16. Laurent, T. Train Ego-Path Detection on Railway Tracks Using End-to-End Deep Learning. *arXiv* **2024**, arXiv:2403.13094 [CrossRef]
17. Zhang, Q.; Yan, F.; Song, W.; Wang, R.; Li, G. Automatic Obstacle Detection Method for the Train Based on Deep Learning. *Sustainability* **2023**, *15*, 1184. [CrossRef]
18. Drizi, H.K.; Boukadoum, M. CNN Model with Transfer learning and Data Augmentation for Obstacle Detection in Rail Systems. In Proceedings of the 2024 IEEE International Symposium on Circuits and Systems (ISCAS), Singapore, Singapore, 19–22 May 2024; pp. 1–5. [CrossRef]
19. Munappy, A.; Bosch, J.; Olsson, H.H.; Arpteg, A.; Brinne, B. Data Management Challenges for Deep Learning. In Proceedings of the 2019 45th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Kallithea-Chalkidiki, Greece, 28–30 August 2019; pp. 140–147. [CrossRef]
20. *EN 50716:2023*; Railway Applications—Requirements for Software Development. Technical Report; CENELEC: Brussels, Belgium, 2023 .
21. Adler, R.; Bunte, A.; Burton, S.; Großmann, J.; Jaschke, A.; Kleen, P.; Lorenz, J.M.; Ma, J.; Markert, K.; Meeß, H.; et al. *Deutsche Normungsroadmap Künstliche Intelligenz*; DIN e. V. und DKE: Berlin, Germany, 2022. [CrossRef]
22. Machida, F. N-Version Machine Learning Models for Safety Critical Systems. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Portland, OR, USA, 24–27 June 2019; pp. 48–51. [CrossRef]
23. Xu, H.; Chen, Z.; Wu, W.; Jin, Z.; Kuo, S.y.; Lyu, M. NV-DNN: Towards Fault-Tolerant DNN Systems with N-Version Programming. In Proceedings of the 2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W), Portland, OR, USA, 24–27 June 2019; pp. 44–47. [CrossRef]
24. Wu, A.; Rubaiyat, A.H.M.; Anton, C.; Alemzadeh, H. Model Fusion: Weighted N-Version Programming for Resilient Autonomous Vehicle Steering Control. In Proceedings of the 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Memphis, TN, USA, 15–18 October 2018; pp. 144–145. [CrossRef]
25. Gujarati, A.; Gopalakrishnan, S.; Pattabiraman, K. New Wine in an Old Bottle: N-Version Programming for Machine Learning Components. In Proceedings of the 2020 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Coimbra, Portugal, 12–15 October 2020; pp. 283–286. [CrossRef]
26. Jass, P.; Abukhashab, H.; Thomas, C.; Woltersdorf, P.; Weber, M.; Conrad, M.; Fey, I.; Schülzke, H. CertML: Initial Steps Towards Using N-Version Neural Networks for Improving AI Safety. *Datenschutz Und Datensicherheit—DuD* **2023**, *47*, 483–486. [CrossRef]
27. Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3523–3542. [CrossRef] [PubMed]
28. Zendel, O.; Murschitz, M.; Zeilinger, M.; Steininger, D.; Abbasi, S.; Beleznai, C. RailSem19: A Dataset for Semantic Rail Scene Understanding. In Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Long Beach, CA, USA, 16–20 June 2019; pp. 1221–1229. [CrossRef]
29. NRK. Nordlandsbanen: Minute by Minute, Season by Season. 2013. Available online: https://nrkbeta.no/2013/01/15/nordlandsbanen-minute-by-minute-season-by-season/ (accessed on 9 March 2025).
30. Michalke, T.; Wüst, C.; Feng, D.; Dolgov, M.; Gläser, C.; Timm, F. Where can i drive? A system approach: Deep ego corridor estimation for robust automated driving. In Proceedings of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), Indianapolis, IN, USA, 19–22 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1565–1571.
31. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2015; Volume 9351, pp. 234–241. .{_}28. [CrossRef]
32. Simonyan, K.; Zisserman, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv* **2015**, arXiv:1409.1556v6. [CrossRef]
33. Gupta, D. Image Segmentation Keras: Implementation of Segnet, FCN, UNet, PSPNet and Other Models in Keras. 2022. Available online: https://github.com/divamgupta/image-segmentation-keras (accessed on 24 February 2025).
34. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv* **2017**, arXiv:1704.04861. [CrossRef]
35. Badrinarayanan, V.; Kendall, A.; Cipolla, R. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2017**, *39*, 2481–2495. [CrossRef] [PubMed]
36. Ziegler, M.; Mhasawade, V.; Köppel, M.; Neumaier, P.; Eiselein, V. A Comprehensive Framework for Evaluating Vision-Based on-Board Rail Track Detection. In Proceedings of the 2023 IEEE Intelligent Vehicles Symposium (IV), Anchorage, AK, USA, 4–7 June 2023; IEEE: Piscataway, NJ, USA, 2023; pp. 1–8. [CrossRef]

37. Willers, O.; Sudholt, S.; Raafatnia, S.; Abrecht, S. Safety Concerns and Mitigation Approaches Regarding the Use of Deep Learning in Safety-Critical Perception Tasks. In *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops, Proceedings of the DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, 15 September 2020*; Casimiro, A., Ortmeier, F., Schoitsch, E., Bitsch, F., Ferreira, P., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2020; pp. 336–350. [CrossRef]
38. Dmitriev, K.; Schumann, J.; Bostanov, I.; Abdelhamid, M.; Holzapfel, F. Runway Sign Classifier: A DAL C Certifiable Machine Learning System. In Proceedings of the 2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC), Barcelona, Spain, 1–5 October 2023; pp. 1–8. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.