Certified control for train sign classification

Jan Roßbach und Michael Leuschel

Article - Version of Record

Science of Computer Programming 246 (2025) 103323

Contents lists available at ScienceDirect

Science of Computer Programming

journal homepage: www.elsevier.com/locate/scico

Certified control for train sign classification

Jan Roßbach [*], Michael Leuschel

Heinrich-Heine-Universität Düsseldorf, Mathematisch-Naturwissenschaftliche Fakultät, Institut Informatik, Universitätsstr. 1, Düsseldorf, 40225, NRW, Germany

ARTICLE INFO

Keywords:
ATO
Artificial intelligence
Formal methods
Computer vision
Autonomous systems

ABSTRACT

Certified control makes it possible to use artificial intelligence for safety-critical systems. It is a runtime monitoring architecture, which requires an AI to provide certificates for its decisions; these certificates can then be checked by a separate classical system. In this article, we evaluate the practicality of certified control for providing formal guarantees about an AI-based perception system. In this case study, we implemented a certificate checker that uses classical computer vision algorithms to verify railway signs detected by an AI object detection model. We have integrated this prototype with the popular object detection model YOLO. Performance metrics on generated data are promising for the use-case, but further research is needed to generalize certified control for other tasks.

1. Introduction

Artificial intelligence (AI) has been increasingly used in various sectors, including transportation [1]. AI is of particular relevance for autonomous driving systems for railways [2], inspired by successful results in other transportation sectors (mainly automotive) [3].

While this technology is of significant economic interest, reliable certification methods are necessary to ensure safe and regulated adoption of these innovations [2]. Traditional verification approaches, such as formal methods, face difficulties due to the probabilistic and opaque nature of AI.

The KI-LOK [4–6] research project aims to address some of these challenges by developing novel verification methods for autonomous AI-based railway systems. Part of this effort is a case study [7] for shunting movements, i.e., controlling a freight train engine in a shunting yard. A formal B [8] model has been developed [7] to analyze the shunting yard environment and ensure the safety of the deterministic steering system through model checking with the ProB [9] model checker.

The safety of the system is conditional on correct results from the AI-based perception system, which is responsible for correct obstacle detection and train signal classification. In this work, we attempt to move towards verification of part of this perception system using a runtime monitor with a certified control [10] architecture. This architecture reduces the part of the system that needs to be formally verified: namely only the certificate checker and not the AI system itself (see Section 2).

In this paper, we focus on a particular subset of the train perception system: the sign classification component. It is responsible for detecting and classifying signs in the shunting yard to ensure safe train movements. Two important signs shown in Fig. 1 are ‘stop’ (Sh0) and ‘track-free’ (Sh1). There are four kinds of outcomes for the perception system:

1. True Positive: the AI correctly detects a sign that is present.

* Corresponding author.
E-mail address: jan.rossbach@uni-duesseldorf.de (J. Roßbach).

https://doi.org/10.1016/j.scico.2025.103323
Received 11 June 2024; Received in revised form 30 April 2025; Accepted 6 May 2025

Available online 13 May 2025

(a) Sh0 (b) Sh1 (c) Wn7

Fig. 1. Some Train Control Signs in a Shunting Yard.

2. True Negative: the AI correctly detects the absence of a sign.
3. False Positive: the AI incorrectly detects a signal that is not there, or detects the wrong signal.
4. False Negative: the AI fails to detect a signal that is there.

A false positive of a 'track-free' signal obviously can have serious safety implications, since it could mean that a stop signal is being ignored and this could lead to a collision. Similarly, a false negative for a 'stop' signal can be equally problematic. Our article only tackles case 3, reducing the false positives. The idea is that the AI perception system will produce in addition to the classification a certificate, which can be checked by a separate (simpler) certificate checker.

- The certificate checker does not rely on AI and is much simpler to certify: it can be tested, validated or even verified using classical technology.
- If the certificate checker is correct, all false positives will be removed.
- On the downside, the certificate checker may fail to accept true positive results, and thus transform true positives into false negatives.

To tackle false negatives, one needs to use other measures. For example, if we know where signs are to be expected, then false negatives can be detected. In [7] the steering system also has access to information about the topology and can thus determine at which locations a sign must be present. If no sign is detected, one can trigger an emergency brake or pretend a Sh0 'stop' sign is there. These measures are outside the scope of this article.

We aim to reduce or eliminate false positives, e.g. incorrectly found signs, by defining a sign-specific ontology and using it for verifying the AI output at runtime. For this we introduce a formal specification in Section 3 and show the performance gains using a prototype implementation (see Section 4) on synthetic data in Section 5.

Our main research question is: can we

1. considerably reduce the false positives of an AI perception system,
2. without a significant increase in false negatives and
3. using a certificate checker that can be certified.

This article extends the work of [11]. It gives more relevant background information and additional details on the evaluation methodology. It also adds a more detailed discussion of the new results and mentions the implications for future work. Since the original publication the data generation has been redone on the basis of project internal videos, which better represent the real world scenarios of the given case study. Originally the images were gathered from the web. We now regenerated the entire dataset with double the amount of images and re-evaluated the monitor, confirming the general trend of the original paper and managed to improve the F-score in several cases.

In Section 2 we introduce the case study and additional relevant background to understand the context of the work and the some relevant concepts in more depth. Then we provide the specification for the later implementation in Section 3 and the details on the implementation in Section 4. Finally, we evaluate how the system performs in Section 5 and discuss the results, before concluding the paper in Section 6.

2. Background & related work

The development of autonomous railway systems has led to the establishment of a categorization system known as Grades of Automation (GoA). This framework defines the level of independence that a train can operate at. The highest degree of automation, GoA4, is defined by full automatic operation [12]. To achieve this, trains must possess two primary functions: Automatic Train Protection (ATP) and Automatic Train Operation (ATO). ATP is responsible for ensuring trains follows safety protocols and avoid collisions, while ATO handles tasks typically performed by human drivers. For both functions, there is a fundamental requirement: the automated system must perform better than a human driver [13]. This means, that each function has an upper limit on its allowed error rate often referred to as the tolerable hazard rate. To demonstrate a convincing safety case, it is essential to provide evidence that error rate will not be exceeded. The accuracy levels of AI models are typically lower than this tolerable hazard rate required for securing autonomous train perception components.

The KI-LOK project, is a collaboration Project that aims to develop a methodology for the certification of AI-based components in train control. In the scope of the project, a case study [7] has been developed by Thales (now Ground Transportation Systems).

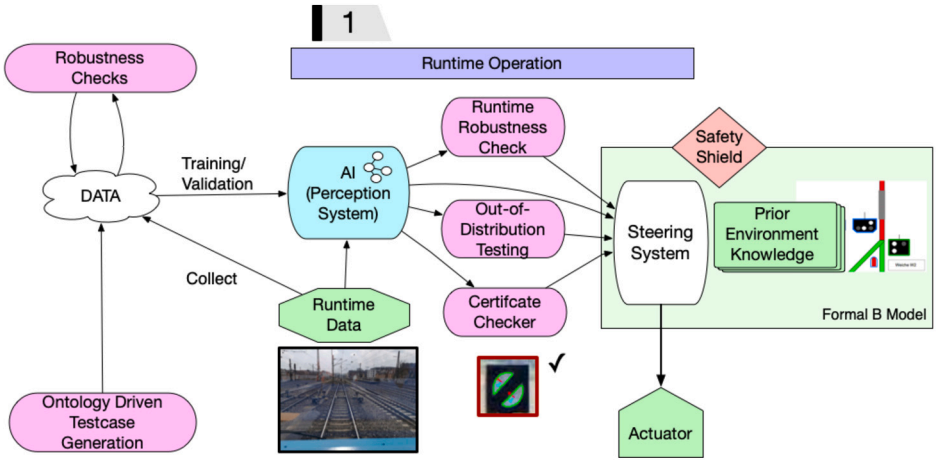


Fig. 2. Various measures to enable certifying AI-based obstacle detection.

The purpose of the case study is to demonstrate and evaluate the feasibility of the projects proposed methodology [6] for AI-based components in train control.

The system includes an AI-based perception system and a deterministic steering system. Fig. 2 shows the interaction between the two systems and the various measures taken to ensure the correctness of the system.

The role of the perception system is to detect and classify obstacles (persons, animals, vehicles) and railway infrastructure elements. The steering system then makes appropriate decisions about moving the locomotive based on that information. The case study contains a set of requirements, which includes the correct detection of a specified list of shunting train signs. In order to increase confidence in the perception system we check the recognized signs with a runtime monitor (certificate checker). This will give stronger confidence that detected signs are correct. In order to safeguard against unrecognized signs we will need to lean on other measures taken by the project, like a thorough environment ontology and systematic test case generation [14].

2.1. Certified control

Certified Control [10] is an architectural framework for the real-time validation of autonomous systems and distinguishes itself from conventional monitoring components, by removing its reliance on independent perception and instead counting on the controller to provide a *certificate* containing all essential information.

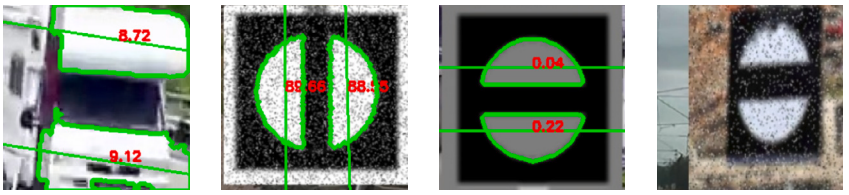
This certificate serves as input for the runtime monitor, which assesses the correctness of the perception against specified criteria. By adopting this approach, the architecture establishes a smaller trusted foundation that can potentially be subjected to a rigorous formal verification process. The controller, which is not included in the *trusted base*, can utilize sophisticated algorithms such as neural networks without needing explicit formal verification. By separating the tasks of generating visual insights and ensuring safety, established verification methods can be used. To accomplish this, a formal acceptance specification for the certificate is necessary to ensure compliance with safety requirements like *the detected lane lines are parallel* or *there are no objects on the track for 100 m*. This reduces the amount of code needing verification and allows the AI components to go unverified.

While the effectiveness of this architecture in lane line detection for regular vehicles is promising [10], its applicability to other autonomous perception tasks such as sign classification and object detection remains uncertain. Therefore, we aim to investigate the applicability and effectiveness of such a certified control architecture in the context of the case studies train control perception system.

2.2. Other related work

Other attempts at verifying an autonomous train perception systems notably include [2,15]. The authors propose a multi-sensor pipeline relying on the statistical independence of the different perception mechanisms to control hazards and ensure suitable model performance. The goal is to show possible ways of certifying according to the ANSI/UL 4600 [16] standard, which provides a framework for integrating AI into fully autonomous systems.

The standard gives practical guidelines and advice for a possible safety case, notably including the entire autonomy pipeline and AI algorithms. We also hope to provide methods to aid with a verification according to this standard, while a full certification is currently out of reach. Other approaches to formal runtime monitor verification of AI systems use safety shields [17]. There have also been proposals for formalizing image specification, including spatial model checking [18] and attempts to formalize vision ontology [19,20].



(a) FP Sh0 accepted (b) FP Sh0 rejected (c) TP Sh0 accepted (d) TP Sh0 rejected

Fig. 3. Visual Examples of Successful and Failing Monitor Checks.

3. Specification and ontology

The selected sign classes for verification are Sh0, Sh1, and Wn7 as depicted in Fig. 1. While these look similar, the semantic content is different. Sh0 means stop and the others signal safe passage. This makes properly locating and distinguishing them a safety-critical issue. To achieve this, we employ an AI object detection system in the controller. Once the AI system has detected a sign, the classification and the part of the image in its bounding box are sent to the verifier. The monitor verifies if this part of the image aligns with the expected ontology. This provides additional confidence, that what has been detected is not a false positive.

It is often challenging to provide a precise formal definition of an image class based solely on its features. Instead, we focus directly on detectable image characteristics. In this context, we can observe that the images include two semi-circles with only orientation as the distinguishing feature. This characteristic feature allows us to define the sign using the contours and orientation angles of the feature.

For a given image tensor I with height h and width w, consider the set of contours (sets of points) denoted as $C(I)$, which are identified by a contour detection algorithm [21]. Let S_0 be the set of images belonging to the Sh0 class. Also define $A : C(I) \to \mathbb{R}^+$ as the area function, which calculates the area of a given contour. Similarly, let $\sigma : C(I) \to \mathbb{Z}^+$ be an orientation function that determines the angle between the contour and the horizontal axis. We can then express membership of an image to one of the classes by considering an image a member of the set S_0 if it contains a pair $(c_1, c_2) \in C(I) \times C(I)$, which full-fills all the following conditions, given some pre-determined error tolerances $\delta_i, i \in \{1, 2, 3, 4, 5\}$[1] and an expected angle a that depends on the class in question.

1. $A(c_1)(1 - \delta_1) \leq A(c_2) \leq (1 + \delta_1)A(c_1)$
2. $(1 - \delta_2)\sigma(c_1) \leq \sigma(c_2) \leq (1 + \delta_2)\sigma(c_1)$
3. $\delta_3 h \leq A(c_i) \leq \delta_4 h, i \in 1, 2$
4. $\delta_3 w \leq A(c_i) \leq \delta_4 w, i \in 1, 2$
5. $c_1 \cap c_2 = \emptyset$
6. $|\sigma(c_i) - a| \leq 90\delta_5, a = 0$

For the remaining two classes, the expected angle a in the final condition varies to 45 for Sh1 and 90 for Wn7. Otherwise, the definitions are identical. The conditions one to six define an Sh0 sign as an image with two contours that have similar angles and orientations. The orientation should be within a certain error threshold. Also, the definition expects, that the areas do not overlap. While ideally, we expect an orientation of zero, variations can occur due to different photo angles. Thus, the inclusion of an error term accounts for this discrepancy in measurement accuracy.

This definition is not flawless and permits the possibility of false positives. This implies that there may be instances where images that do not depict the intended sign could potentially be accepted (see Fig. 3a). However, incorporating this check reduces the likelihood of such occurrences compared to those without it. The stringency of the monitoring process needs to be weighed against the decrease in true positives to strike a suitable balance. Adjustments can be made by selecting appropriate δ values within certain limits.

Defining such a specification now allows us to define a requirement to prove the monitor correct, assuming that the underlying contour detection algorithm is correctly implemented.

REQ: The implementation accurately verifies whether an image meets the ontology requirements of a specific class.

While this work provides a verification of REQ for the given implementation in the next section, such a verification can trivially be obtained in any serious production environment for a given monitor. One only has to show that the implementation specifies the correct values and does the correct conditional comparisons, which can be done via typical program verification tools such as *Z3* [22].

4. Implementation

While the following implementation is not yet verified in terms of *REQ*, we aim to do so in future work. Here we provide a prototype, which is developed enough to indicate the potential usefulness of such an implementation. Given an image and an expected class,

[1] In the prototype implementation the tolerance values used were $\delta_{1,2,5} = 0.2$, $\delta_3 = 0.1$ and $\delta_4 = 0.3$.

Table 1
Raw numbers for Models on Generated Data.

Model	Detected	TP	FP
n	58003	40563	17440
s	52096	49052	3044
m	51733	41598	10135

(a) Results without Monitor

Model	Detected	TP	FP
n	58003	32902	1
s	52096	39843	1
m	51733	37793	0

(b) Results with Monitor

Table 2
Model Metrics on Generated Data (values rounded to two decimal places).

Model	Precision	Recall	F_1
n	0.69	0.71	0.70
s	0.94	0.68	0.79
m	0.80	0.64	0.71

(a) Results without Monitor

Model	Precision	Recall	F_1
n	1.00	0.66	0.79
s	1.00	0.64	0.78
m	1.00	0.62	0.77

(b) Results with Monitor

it either validates or rejects the image. We then integrated it with a YOLOv8 object detection model and measured the influence on common performance metrics (see. Table 2b). In the following sections, we present details on the implementation of the controller and monitor components.

4.1. Controller

The AI-based controller component is responsible for finding and correctly recognizing the sign in the image, which is a task that is out of reach for traditional computer vision algorithms but can be effectively solved by advanced deep learning models, such as the popular YOLO [23] architecture. In contrast to image classification, the task of object detection has to also find the object's location in the image. It will return the bounding box of the object or objects found in the given image and a corresponding class prediction.

Our controller implementation uses a fine-tuned YOLOv8[2] model. We trained three model variants on a manually labeled custom sign-detection dataset [24]. These were the nano, small and medium (abbreviated n, s and m in Table 1 and Table 2), versions of the model with 3.2M, 11.2M and 25.9M parameters respectively. The training was done for 200 epochs with a batch size of 16. They achieved mAP50 values of 0.827, 0.90 and 0.93 on the test set.

From the model results the controller generates a *certificate* – in the sense of certified control (see Section 2) – consisting of the following components:

1. The original image itself.
2. The assigned class result.
3. The bounding box, represented as a tuple in the format (x, y, w, h), with values normalized to fit the dimensions of the image.

This Python object is then given to the monitor. In a production implementation, it would be preferable to serialize and send this data to a statically typed version of the monitor for optimal security.

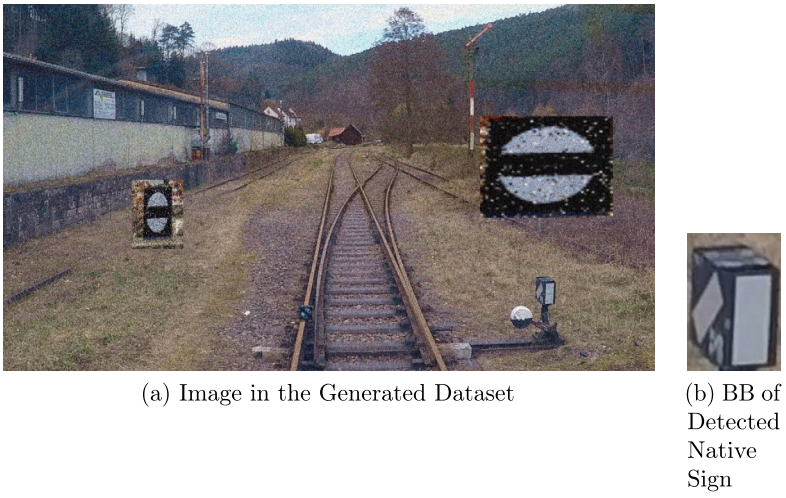
4.2. Monitor

The monitor is responsible for verifying the correctness of the controllers class predictions for each object found in the given image. It takes the *certificate* and verifies that the part of the original image inside the given bounding box matches the specified ontology of the predicted class result. If the class prediction is correct and the bounding box is reasonably accurate, we expect the monitor to accept the image. In case of a false positive model prediction it is very unlikely, but not impossible, that the monitor will accept the *certificate*.

The implementation utilizes Python's OpenCV [25] library to apply simple, well-tested computer vision algorithms to the given images. To start with, the bounding box image is cropped from the original image and re-sized to 206x206. It is then converted to grayscale to facilitate contour detection. After, the contour detection is performed using OpenCV's findContours function. Subsequently, a filtering process is applied to the contours to ensure their area falls within the specified size boundaries (refer to Section 3). Then, the area and orientation of the detected contours is calculated to determine if some of them fit the requirements for the ontology. The area of each contour is extracted using an available function within OpenCV. In addition, we utilize OpenCV once again by fitting a line through each contour as a means of determining its orientation. With that, we can calculate the orientation using the following equation.

$$\sigma(c) = \frac{180 \arccos(\vec{e_1} \cdot \vec{v})}{\pi |\vec{v}|}$$

[2] https://github.com/ultralytics/ultralytics.



(a) Image in the Generated Dataset

(b) BB of Detected Native Sign

Fig. 4. Generated Data Example. (The image is slightly pixelated for copyright reasons.)

Next, we evaluate the remaining contours in pairs to determine if they satisfy the similarity conditions for area and orientation (refer to Section 3 for details). If a pair is found that meets these conditions, we verify if its orientation aligns with the expected orientation for the corresponding class. If it does, the monitoring system considers this as a valid certificate. However, if any of these criteria are not met, the certificate will be rejected.

5. Experiments, results & discussion

In contrast to the automotive field, which benefits from large-scale image datasets like KITTI [26] for efficient object detection model evolution using road scene images, the railway industry faces limitations in terms of relevant datasets [27]. Recently, interesting multi-sensor benchmark datasets [28] have started to emerge, but do not fit our particular use case.

This lack of labeled, high-quality data poses a challenge when it comes to training and validating AI-based systems for this particular case. When evaluating the performance of the prototype, we have to confront this lack of data in the field. Since the relevant publicly available datasets do not cover the classes in question, we resort to custom labeling for training and a data generation approach for the evaluation of the system.

5.1. Evaluation data generation

For the image generation of the evaluation data, we chose a small number of base images of the signs in question, which are put through different random perturbation combinations and then pasted in random amounts – one to four – onto images from train footplate rides, gathered by industry project partners.

We generated 45472 unique images this time, containing 74935 signs in total. There are up to four signal per picture, which is typical of a shunting yard. Fig. 4a shows one image that has been generated by this process. This image happens to contain a real train switch signal, that happens to be confused for an Sh1 Sign by the YOLO model (see Fig. 4b) and rejected by the monitor. However; such native signals in the source images have been ignored for the later analysis, because they are not labeled.

The following perturbations were applied to the base images before cutting them onto the video frames:

1. Horizontal Flip
2. Gaussian Noise (Salt and Pepper with Levels of 0.05 and 0.075)
3. Scaling (To square images of 50, 100, 213, 416 and 832 px and back)
4. Blur (normalized box filter with kernel sizes 3, 5, 7)
5. Brightness change (levels 0.5,1.5)

In Fig. 3 we see examples of these images with monitor visualizations applied. It shows cut YOLO bounding boxes with the contours, lines and corresponding orientations detected by the monitor. Fig. 3a shows one of the few remaining false positives.

The image fits all the defined criteria of the S_0 ontology for these δ values but is not actually of that class. Given stricter tolerances (e.g. $90\delta_5 < 8$) this mistake would not occur.

For the evaluation, we used the trained YOLOv8 model (see Section 4.1) on the generated images to detect the signs. For each detection result, we determine the closest label – in terms of L2 distance of the two box centers – to the detection result. If the detected class matches the label class, we count it as a true positive, otherwise a false positive. This method ignores how well, or poorly, the detected bounding box matches the label bounding box, but we omit this for simplicity here.

After determining the status of the detection as true positive or false positive, we crop the part of the image inside the detected bounding box, and send it to the monitor (see Section 4.2). If the original result was a true positive, but the monitor rejected it, we count it as a false positive in Table 1b. Otherwise it stays a true positive. If the original result was a false positive, and the monitor accepted it, it stays a false positive. Otherwise it becomes a true negative.

5.2. Results

The results of the evaluation can be found in Table 1 and Table 2. Table 1 shows the raw results of how many signs of the 74935 available signs were detected by the model and how many true positives and false positives were detected with (in Table 1b) and without (see Table 1a) the monitor. In Table 2 we see how those numbers translate to several relevant performance metrics. Precision is the ratio of true positives to the number of model detections. Meaning a ratio of how many of the objects detected by the model turn out to be accurately detected and correctly classified. Recall is the ratio of true positives to the number of actual positives, meaning how many of the actual ground truth signs in the dataset were detected by the model. The F-score is the harmonic mean of precision and recall.

In terms of runtime performance, the monitor checks a certificate in approximately 0.7 ms on an Intel i5-12600K processor. In comparison, the inference of the YOLOv8 model will range from 2 ms – for the nano model variant – to 8 ms for the m version. This means that the performance overhead is likely not a major concern in a production environment.

5.3. Discussion

The results show that the monitor is able to successfully prevent almost all of the safety-critical false positives, leading to a precision of around 99.9982%. We also observe the expected drop in recall, due to the fact that the monitor rejects true positives. This is an essential trade-off of the approach. If the model does detect a sign correctly, but it can not be verified, for instance because the contour detection algorithm fails or the features are slightly outside of the expected range given in Section 3, the monitor will add a false negative and the recall will drop accordingly. For instance, the n model in Table 1 shows 40563 true positives before and 32902 after the monitor was applied. This implies that 7661 new false negatives were added by the monitor.

However, the results still show an overall improved F-score in some of the models. This is due to the fact, that the drop in recall is small enough and these models have a lower precision on the dataset, making the jump to 1.0 in precision more significant for the F-score. In general, a model with better precision will benefit the least from the monitor in terms of raw performance gains. It should however be said, that the performance improvements in these results are not the purpose of the work and are not expected translate to actual systems. Instead, the monitor is only meant to give additional assurance about safety-critical properties and avoid potential hazards caused by false positives.

The model also still shows a high number of false negatives, e.g. signals that were not detected by the model, which is likely due to the difference in training and evaluation data. Indeed, the evaluation data was artificially generated and has a quite different distribution from the real-world training data. Hence, we did not expect the trained model to perform to a high level. But this is not a problem: the contribution of this paper is not the model, but the runtime monitor that surrounds it.

Additionally, the data permutations in the evaluation data are relatively strong to simulate rough edge cases and occlusions. Fig. 5 shows an example of how a low brightness permutation in the evaluation data can cause a false negative. The sign in the bottom left corner is not detected because the features are barely visible. This is done intentionally to provoke a large number of mistakes on the part of the model, so that we can observe changes in performance due to the monitor. The goal of the work is to present an implementation for a runtime monitor that reduces the number of false positives in a given AI implementation, and not to show that the model does not make mistakes or verify that it is better than a human driver in any realistic scenario. The monitor however can not verify a non-existing AI result, meaning the high number of false negatives have to be addressed by other techniques.

This could be done with model improvements through longer training or more training data, or through other parts of the system. Other external techniques could include providing the system with prior knowledge on when signs are expected on the track. This would allow the system to detect when a false negative has occurred and respond appropriately.

Because of the monitor's simplicity it can potentially be formally verified, assuming the underlying computer vision algorithms are at some point proven to be correct. As previously mentioned in Section 3 they are widely used and therefore "proven-in-use" but not yet formally verified. However, this is currently a more realistic goal than verifying the AI algorithms themselves [10].

6. Conclusions and future work

This work presents a novel approach for improving the reliability of perception systems for autonomous train systems. It demonstrates the potential of the certified control architecture in this context. The architecture employs a monitoring mechanism that rejects detections failing to meet specified criteria, thereby almost entirely eliminating false positive detections. To test the approach we created a specific implementation for a given case study on a generated dataset, demonstrating significant improvements in precision and F-score compared to the original model without monitoring. Our results suggest that this technique has the potential to produce more reliable perception systems for railway systems.

For future work, we aim to extend the monitor to detect other types of objects and create a more advanced implementation in a type safe language and formally verify it along the lines of REQ. This implementation should then be tested in real production environments in order to determine if the observed performance gains hold up in practice.

Fig. 5. Example of strong augmentation causing a false negative: the Sh0 sign in the lower left of the image is not detected by the Yolo model due to strong pixelation.

We are also focusing on more broad application of the method for different real-world use cases. One of them being the verification of train track segmentation task in context of a train odometry case study. We aim to develop tools for working with this technology including integration with popular open source visualization tools.

CRediT authorship contribution statement

Jan Roßbach: Writing – review & editing, Writing – original draft, Visualization, Software, Methodology, Investigation, Conceptualization. **Michael Leuschel:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Conceptualization.

Funding

This part of the KI-LOK project funded by the “Bundesministerium für Wirtschaft und Energie”; grant # 19/21007E.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Michael Leuschel reports financial support was provided by Bundesministerium für Wirtschaft und Energie. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] D. Ristić-Durrant, M. Franke, K. Michels, A review of vision-based on-board obstacle detection and distance estimation in railways, Sensors (2021), https://doi.org/10.3390/s21103452.
[2] J. Peleska, A.E. Haxthausen, T. Lecomte, Standardisation considerations for autonomous train control, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation. Practice, Springer Nature, Switzerland, 2022, pp. 286–307.
[3] R. Tang, L. De Donato, N. Besinovic, F. Flammini, R.M. Goverde, Z. Lin, R. Liu, T. Tang, V. Vittorini, Z. Wang, A literature review of artificial intelligence applications in railway systems, Transp. Res., Part C, Emerg. Technol. 140 (2022) 103679, https://doi.org/10.1016/j.trc.2022.103679.
[4] G. Hemzal, T. Strobel, J. Großmann, B.-H. Schlingloff, M. Leuschel, S. Sadeghipour, J. Firnkorn, KI-LOK – a joint test procedure project for AI-based components used in railway operations, Signal + Draht (2021).
[5] G. Hemzal, T. Strobel, M. Leuschel, J. Großmann, D. Knoblauch, M. Kucheiko, N. Grube, R. Krajewski, KI-LOK - Ein Verbundprojekt über Prüfverfahren für KI-basierte Komponenten im Eisenbahnbetrieb, Signal + Draht (2023).
[6] J. Roßbach, O. De Candido, A. Hammam, M. Leuschel, Evaluating ai-based components in autonomous railway systems, in: A. Hotho, S. Rudolph (Eds.), KI 2024: Advances in Artificial Intelligence, Springer Nature, Switzerland, Cham, 2024, pp. 190–203.
[7] J. Gruteser, D. Geleßus, M. Leuschel, J. Roßbach, F. Vu, A formal model of train control with ai-based obstacle detection, in: B. Milius, S. Collart-Dutilleul, T. Lecomte (Eds.), Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification, Springer Nature, Switzerland, 2023, pp. 128–145.
[8] J. Abrial, A. Hoare, The B-Book: Assigning Programs to Meanings, Cambridge University Press, 2005.
[9] M. Leuschel, M. Butler, ProB: a model checker for B, in: Proceedings FME, in: LNCS, vol. 2805, 2003, pp. 855–874.
[10] D. Jackson, V. Richmond, M. Wang, J. Chow, U. Guajardo, S. Kong, S. Campos, G. Litt, N. Aréchiga, Certified control: an architecture for verifiable safety of autonomous vehicles, CoRR, arXiv:2104.06178, 2021, https://doi.org/10.48550/arXiv.2104.06178.
[11] J. Roßbach, M. Leuschel, Certified control for train sign classification, in: Proceedings Fifth International Workshop on Formal Methods for Autonomous Systems 395 (2023) 69–76, https://doi.org/10.4204/eptcs.395.5.
[12] N. Bešinović, L. De Donato, F. Flammini, R.M.P. Goverde, Z. Lin, R. Liu, S. Marrone, R. Nardone, T. Tang, V. Vittorini, Artificial intelligence in railway transport: taxonomy, regulations, and applications, IEEE Trans. Intell. Transp. Syst. 23 (9) (2022) 14011–14024, https://doi.org/10.1109/TITS.2021.3131637.

[13] J. Athavale, A. Baldovin, M. Paulitsch, Trends and functional safety certification strategies for advanced railway automation systems, in: 2020 IEEE International Reliability Physics Symposium (IRPS), 2020.
[14] J. Grossmann, N. Grube, S. Kharma, D. Knoblauch, R. Krajewski, M. Kucheiko, H.-W. Wiesbrock, Test and training data generation for object recognition in the railway domain, in: P. Masci, C. Bernardeschi, P. Graziani, M. Koddenbrock, M. Palmieri (Eds.), Software Engineering and Formal Methods. SEFM 2022 Collocated Workshops, Springer International Publishing, Cham, 2023, pp. 5–16.
[15] J. Peleska, F. Brüning, M. Gleirscher, W. ling Huang, A stochastic approach to classification error estimates in convolutional neural networks, CoRR, arXiv: 2401.06156, 2023.
[16] U.L. Inc, 4600 standard for evaluation of autonomous products, Tech. Rep., Underwriters Laboratories Inc., 2020.
[17] B. Könighofer, F. Lorber, N. Jansen, R. Bloem, Shield synthesis for reinforcement learning, in: T. Margaria, B. Steffen (Eds.), Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles, Springer International Publishing, Cham, 2020, pp. 290–306.
[18] V. Ciancia, D. Latella, M. Loreti, M. Massink, Model checking spatial logics for closure spaces, Log. Methods Comput. Sci. 12 (4) (2016), https://doi.org/10.2168/LMCS-12(4:2)2016.
[19] D. Porello, M. Cristani, R. Ferrario, Integrating ontologies and computer vision for classification of objects in images, in: Proceedings of the Workshop on Neural-Cognitive Integration in German Conference on Artificial Intelligence, 2013, pp. 1–15.
[20] R.I. Minu, K.K. Thyagharajan, Semantic rule based image visual feature ontology creation, Int. J. Autom. Comput. 11 (5) (2015), https://doi.org/10.1007/s11633-014-0832-3.
[21] S. Suzuki, K. be, Topological structural analysis of digitized binary images by border following, Comput. Vis. Graph. Image Process. 30 (1) (1985) 32–46, https://doi.org/10.1016/0734-189X(85)90016-7.
[22] L. de Moura, N. Bjørner, Z3: an efficient smt solver, in: C.R. Ramakrishnan, J. Rehof (Eds.), Tools and Algorithms for the Construction and Analysis of Systems, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 337–340.
[23] J. Redmon, S.K. Divvala, R.B. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE Computer Society, Los Alamitos, CA, USA, 2016, pp. 779–788.
[24] KILOK, Sign detection dataset, https://universe.roboflow.com/kilok/sign-detection-4oqe4, may 2023.
[25] Itseez, Open source computer vision library, https://github.com/itseez/opencv, 2015.
[26] A. Geiger, P. Lenz, R. Urtasun, Are we ready for autonomous driving? The kitti vision benchmark suite, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 3354–3361.
[27] M.J. Pappaterra, F. Flammini, V. Vittorini, N. Bešinović, A systematic review of artificial intelligence public datasets for railway applications, Infrastructures 6 (10) (2021), https://doi.org/10.3390/infrastructures6100136.
[28] R. Tilly, P. Neumaier, K. Schwalbe, P. Klasek, R. Tagiew, P. Denzler, T. Klockau, M. Boekhoff, M. Köppel, Open sensor data for rail 2023, https://doi.org/10.57806/9MV146R0, 2023.