Validated autonomous train perception using interpretable machine learning

Luca Jourdan, Mohamed Sallak, Walter Schön, Benjamin Quost, Yohan Bouvet

▶ **To cite this version:**

Luca Jourdan, Mohamed Sallak, Walter Schön, Benjamin Quost, Yohan Bouvet. Validated autonomous train perception using interpretable machine learning. Lambda Mu 22 - Congrès de maîtrise des risques et de sûreté de fonctionnement, Oct 2022, Paris Saclay, France. hal-03878419

HAL Id: hal-03878419
https://hal.science/hal-03878419

Submitted on 29 Nov 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Institut pour la Maîtrise des Risques
Sûreté de Fonctionnement - Management - Cindyniques

Validated autonomous train perception using interpretable machine learning

*Perception validée du train autonome par des modèles d'apprentissage interprétables

Luca Jourdan*, Mohamed Sallak†*, Walter Schön†*, Benjamin Quost†* and Yohan Bouvet†

* Railenium
Valencienne, France
luca.jourdan@railenium.eu

† Université de technologie de Compiègne,
CNRS, Heudiasyc
(Heuristique et Diagnostic des Systèmes Complexes),
CS 60 319 - 60 203
Compiègne, Cedex
mohamed.sallak@hds.utc.fr, walter.schon@hds.utc.fr, benjamin.quost@utc.fr, yohan.bouvet@hds.utc.fr

Résumé—Le développement de trains autonomes requiert le développement de nombreuses fonctions pour détecter, percevoir, reconnaître et prendre en considération les situations dangereuses. Une de ces fonctions est la capacité à détecter un obstacle sur la voie. Le choix a été fait, au sein du projet auquel nous travaillons, d'utiliser le deep learning pour effectuer cette tâche.
Dans ce papier, nous proposons d'utiliser un arbre de décision, une méthode issue de l'apprentissage machine, pour combiner les sorties d'un classifieur basé sur un réseau de neurones avec des informations extérieures tel que la météo afin d'obtenir un système présentant de meilleures performances. L'interprétabilité des arbres de décision nous permet de faire une analyse de sûreté de fonctionnement du travail proposé. Cette analyse de sureté de fonctionnement est une première étape vers la certification d'un système basé sur l'intelligence artificielle pour répondre à une fonction sécuritaire.

Abstract—The development of autonomous trains will require the development of many functions to detect, perceive, recognize and take into consideration hazardous situations. One of those functions is the ability to detect obstacles on the track. Due to the high accuracy of deep learning, the choice has been made, in the context of our work, to use it in order to characterize the environment.
In this paper, we propose to use a decision tree, a machine learning method, to combine the probability output of a neural-network classifier with external information such as the weather to obtain a better performing decision system in terms of availability and safety. The interpretability of the decision tree allow us to make a safety analysis of the proposed work which is a first step toward the certification process of an AI-based system used in a safety-related function.

This research work was granted public funds within the scope of the French Program "Investissements d'Avenir".

Keywords—Autonomous train, GoA4, classification tree, machine learning, validation.

I. Introduction

A. Context

Autonomous trains can drive themselves and make decisions using both their own sensors and potential track-side sensors without human intervention. This approach differs from an automatic metro, which already exists, but which, unlike a train, drives only in a closed and controlled environment. The degree of automation of an autonomous train is indicated by the grade of automation (GoA) specified by the standard IEC 62290-1 [19]. The GoA range from GoA0 (no automation at all) to GoA4 where no personnel is needed onboard the train. An autonomous train should be more cost-effective, less prone to failure, and more energy (Huang et al. [22], [24], [25], Yin et al. [23]), and time efficient than a human driver (Huang et al. [22]). Following the recent success in GoA2 train operations where the autonomous system drives between stations in nominal situations more and more projects such as the Australian AutoHaul project [20] or the French's TFA and TASV project [21] aim to develop GoA4 trains.

The TFA (Train de Fret Autonome, "autonomous freight train") project, initiated by the SNCF (Société Nationale des Chemins de Fer, "french's national society of railway") regrouping the SNCF, Railenium, a technological research

institute, academic partners (UTC and LAMIH) and industrials (Alstom Transport, Altran, Apsys, Hitachi Rail STS), aim to create a GoA4 autonomous freight train prototype by 2023.

To be able to run autonomously in an open, uncontrolled environment, the autonomous train must be able to sense and characterize its environment and, from those perceptions, take decisions in hazardous situations. As with autonomous cars (Kuutti et al. [2]) the need to characterize an open environment led the TFA project to use deep learning to characterize the environment. The usage of machine learning (ML) in railway applications is rare and, as far as we know, there are only a few examples of such usage. Most of the examples that we do know of, do not come to replace a human but to assist him in its function during the development process or to fulfill functions that were not met before. A good example of the usage of ML to assist humans is to help to assess the risk of railway accidents [3] or to assess the safety of critical software used in railway transport [4]. An example of the usage of AI to fulfill functions not fulfilled by humans can be found in the work of Fantini et al [5] which used AI to detect rock falling on the track or the work of Xu et al [6] which used ML for smart power allocation in railway. However, the usage of machine learning to replace humans operator is, to the best of our knowledge, only attempted in autonomous driving projects such as in TASV (Mahtani et al. [7]). The main reason why the usage of ML to replace a human operator in railway operations is so rare is due to the concern that it raises about the safety impact and that, for this reason, it is usually advised against by safety standards such as the EN 50128 [10] for railway or the more general IEC 61508-7[2]. However, due to the increasing demand to use AI, the EPSF, a French public railway safety institution, is working to define the necessary condition to authorize systems using AI and more specifically machine learning algorithms. Furthermore, during the definition in 2020 of a ”smart and durable mobility” strategy, the European commission insisted on the need to stimulate innovation, the usage of data, and AI to serve smarter mobility.

B. Motivations and objectives

Annex A of the software certification standard EN 50128 lists techniques of development associated with the requirements level for each safety integrity level (SIL). This requirement ranges from mandatory to Not Recommended. Table A.3 – Software Architecture, lists the technique “Artificial Intelligence – Fault Correction” as not-recommended for safety-related functions (SIL1, SIL2, SIL3, and SIL4 functions). While the norm limits the perimeter of this requirement to the tasks of ”fault correction” it is usually acknowledged that this restriction is only due to the limitation of AI at the time when this rule has been written and is to be applied for any safety-related function. In the case where this non-recommendation is not respected, the norm demands a “... rationale for using alternative techniques ...”. From the references used to justify this non-recommendation [24][25] we can deduct that the main argument for the non-recommendation of AI is its lack of interpretability. For this reason, we believe that a highly interpretable AI system on which we are able to make a safety analysis should be acknowledged as potentially usable in a safety-related function.

In the TFA project, deep learning, a subset of AI, is used to characterize the perceptions coming from the sensor. The characterized perceptions are then used to take decisions in the presence of obstacles. The decision process actually used by trains conductor has been rationalized through an SNCF document and adapted to an autonomous system through the TFA project. As of now, trains conductor thinks in absolute term and not in probabilistic term and, for this reason, the rules that they used and that the autonomous train will use can be interpreted as a binary rule-based system. For this reason, it is expected that the decision process will only use binary input and then apply the rule corresponding to the hazardous situation. The problem with such a system is that, unlike a human perception, the output of a perception based on a neural network is an uncalibrated probability associated with the predicted class (Guo et al. [18]). For those reasons, the perception system will have to choose at which probability levels a detection should be considered as ’true’ or ’false’.

As seen in Fig. 1, in such a system, choosing the right threshold to binarize the output of the neural network is the same as taking the decision to react or not to a perception. If the threshold is put too low, the system will not disregard potential hazards, but it will lead to many false positives which will impact the performance of the system. On the other hand, a threshold put too high will lead to disregarding potential hazards and will lead to disastrous situations. Choosing the right threshold is equivalent to choosing the right compromise between the safety and the performance of the autonomous train. We believe that such a decision should be taken by legislators, train operators, railway companies, and civil society and, for this reason, we will not try to define the right compromise. Instead, in this paper, we will address the issue of decreasing the number of false positives without negatively impacting the false negative rate. The reason for this choice is that, as of now, the system does not exhibit instances of failing to detect obstacles in videos and the false negatives only come from the fact that the system perceive objects at a short range compared to humans. We will then show that our work can be analyzed using existing safety analysis methods. To make this demonstration, we choose to analyze the remaining risk when using our algorithm using a fault tree analysis.

While standards discourage the usage of AI in a safety critical function, we think that it will be useful for the purpose of choosing the ”good” threshold. For this reason, we propose in this paper to use ML, to fix the threshold value of the perception system. The usual approach to obtain a binary

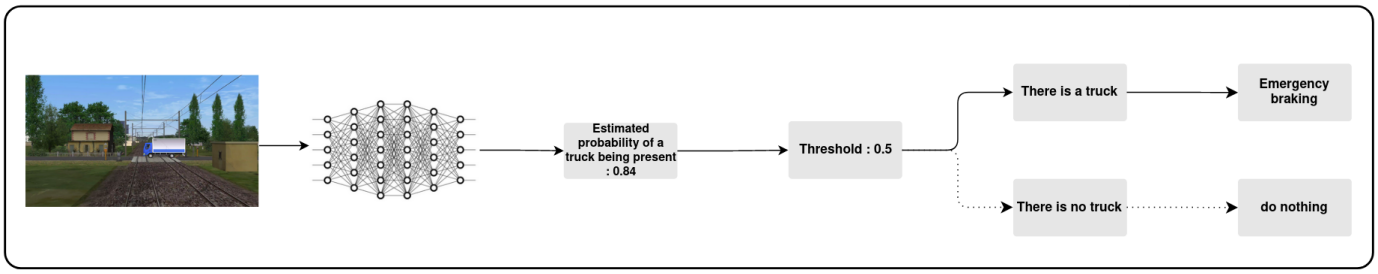

Figure 1. Decision process using a threshold

output from deep learning consists in choosing a threshold that will be used to classify each instance. Such a threshold often defaults to the value of 0.5 but can be changed to optimize the ROC-curve, the precision-recall curve, or any other pertinent metrics. However, such approaches do not take into account other information which might change the threshold, such as the weather. For this reason, we will not use one of those techniques and will use a classification tree, a ML technique, to learn which threshold should be used depending on weather and light conditions. A classification tree, such as defined by Breiman et al. [12], is a tree structure in which a node represents a condition on a feature and a leaf represents a class label (see Fig. 2 for a simple example). Such trees are used in machine learning as a predictive modeling approach and have the advantage, over many other ML algorithms, of being easily interpretable. Due to this interpretability, they have been used in work on safety-related functions such as autonomous car driving (Schmidt et al. [13]). However, the safety demonstration of decision tree usage is, to the best of our knowledge, only addressed on the prism of constraint violation (Schmidt et al. [13]). While this approach is pertinent for some usage, it is impractical for problems when we are unable to express clear constraints that should not be violated such as in an environment characterization's setting. For this reason, we propose to work on the analysis of the induced risk of our approach using fault tree analysis.

II. Background

A. Existing system

As of now the perception system on which we work is composed of cameras capturing the environment, a YOLO (Redmon et al. [1]) based classifier, and a threshold fixed at the value of 50% for each class. The YOLO classifier (You Only Look Once), a deep learning classifier model using convolutional neural networks to convolve multiple filters (also called kernels) to produce feature maps can detect, position, and classify objects on images. Each YOLO detection is associated with a probability. As described in 1, the threshold is in charge of transforming those probabilities into binary values.

According to the rules that our decision system should follow, the train should use the emergency brake when any kind of obstacle is on the track on which the train rolls. For this reason, differencing between the different kinds of existing obstacles is not important to choose to apply or not the emergency brake (it might however be a necessity to consider complementary actions).

The existing system is highly performant at close distances but lacks performance at higher distances. For this reason, the system produces a lot of false negatives for images of obstacles at high distances of the train but has a low number of false negatives on wholes videos sequences. More problematic, though, the system produces, when looking at the probability per hour, a high rate of false positives ultimately hurting the train's performance.

B. Decision tree and gridsearch

As seen in Fig. 2, a tree is a machine learning model with a tree-like structure. In this structure, an internal node corresponds to a test on one of the features, which splits a region of the input space into two regions. In the classification step, a leaf is reached after visiting a series of nodes: the decision associated with it corresponds to the majority class in the subset of training instances falling into the corresponding region. During the learning stage, the algorithm needs to determine when to split the dataset, which variable to use for each split, and with which value.

As with many ML methods, decision tree's performances can vary greatly depending on the hyperparameters used. Of those hyperparameters, the more important ones are the following: the criterion, which is the metric used to decide which split to choose, the maximum depth of the tree, and the maximum number of leaf nodes that impacts the ability of the tree to learn and to overfit, the max feature which indicates how many features should be considered, as most, when splitting a node and the class weight which allows putting more or less importance on different classes.

As the number of hyperparameter combinations increases rapidly with the allowed space search, it is highly impractical to test all of them by hand. As an answer, gridsearch allows us to compare the result for all of those hyperparameters in one go. The idea behind gridsearch is simply to test every possible combination of parameters. For this reason, the space search should still be limited.

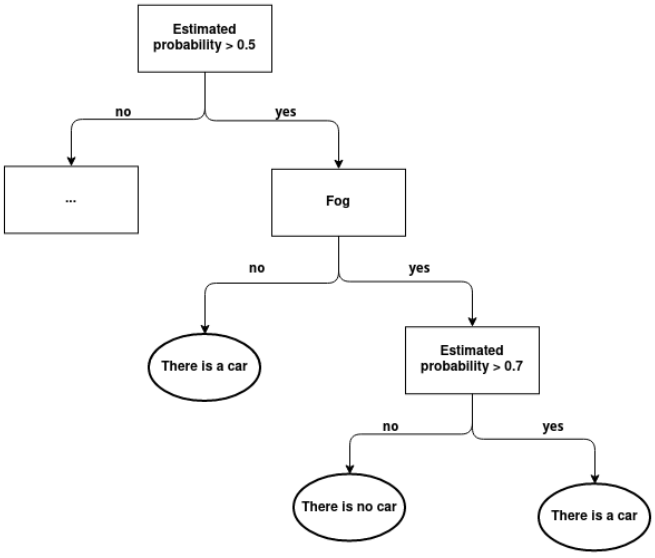

Figure 2. Example of a learned decision tree

C. Fault tree

Fault tree analysis is a visual, deductive failure analysis in which an undesired event of a system is analysed. This analysis, performed using boolean logic to combine lower-level events, can, among other usages, be used to calculate the probability of an undesired event. The top event of a fault tree, called the "undesirable event", is the event that we which to avoid or, in the case of a quantitative analysis, that we which to know the probability of occurrence. The leaves of the fault tree represent the basic events and the intermediate nodes between the leaves and the top event are logical gates including (but not limited to) AND and OR gates. By propagating and combining probabilities or probabilities intervals from low-level events to upper-level events one can calculate the probability or a probability interval of occurrence of the undesirable event. During this calcul, an OR gate correspond to an addition of each corresponding leaves probabilities and an AND gate correspond to a multiplication of each corresponding leaves probabilities.

III. Methodology

A. Decision tree

As seen in the example of a decision tree, interpreting the decision made by a tree is straightforward; it will remain understandable by a human expert as long as the tree has a reasonable depth. This parameter also limits the understanding of the overall behaviour of the tree, since the number of decision regions grows exponentially with it. Last, it should be stressed that large trees tend to overfit the data. For all of these reasons, we considered trees with a maximum depth of 10.

In our context, the variables used to train the decision tree (and consequently used in the sequences of tests required to classify an instance) are the outputs of the perception system (both the detected class and the associated probability), the luminosity conditions, and the weather since these latter arguably change the output distribution of the perception system and should therefore be taken into account by the decision process.

We used the scikit-learn (Pedregosa et al. [14]) implementation of decision trees. To train the tree from both numerical and categorical inputs, we re-encoded categorical data (weather, detected class, and day/night conditions) using one-hot encoding.

Finally, as previously discussed, we used gridsearch to choose the best hyperparameters of the decision tree for our problem with 3-fold cross-validation. While training and testing a tree take seconds with a dataset of the size used in this work, a gridsearch can cover millions of parameters combination and, for this reason, the search has been limited to a subset of the possible search space allowing to limit the learning phase to a few hours on a CPU running at 4.7 GHz. This search has been done on the split criterion (which determines when and how to split a node), the maximum depth of the three, the maximal number of features to be tested for each new split, and the maximum number of leaf nodes (which can have a great impact on overfitting), and the classes weights, which indicate the relative importance of the classes.

Table I
GRIDSEARCH PARAMETERS AND RESULTS

	criterion	maximum depth	max features	class weight	maximum number of leaf nodes
Search space	gini, entropy	[3;10]	{'sqrt', 0.2, 0.4, 0.6, 0.8, 1}	{[1,5], [1,5]}	{5, 10, 15, 20, 25, 30, 35, 40, 50}
Best parameters	entropy	9	0.4	{4,3}	40

B. Dataset

Due to the difficulty to obtain enough real-life images of a diversity of obstacles in different meteorological situations, we used images coming from Heudiasyc's freight train 3D simulator [15]. This simulator allowed us to play and save videos scenario containing obstacles of the different classes with different weather.

Using this simulator, we created a dataset of 24 videos for a total of 800 seconds at 30fps. Those videos combine a total of 30 situations with obstacles from which we extracted 5095 images containing at least one obstacle. Combined with 20229 images without obstacles, the obtained dataset contains 25324 images. Each kind of obstacle (humans, animals, cars, buses, and trucks) is present on both snowy, foggy, nights with good meteorological conditions and days with good meteorological conditions environments. A few samples of

Table II
REPARTITION OF THE CONSTITUTED DATASET

Weather's condition	clear (neither snow, fog or night)	Snow	Fog	Night
No obstacle	9239	3909	1258	4670
Obstacle	2652	1079	2411	106

the images contained in our dataset can be visualized in Fig. 3 and the repartition of the dataset between day, nighty, snowy and foggy environments can be found on II. The ground truth has been obtained by manual annotation and refers to the fact that a human operator is able or not to detect an obstacle. In cases where a human is unsure of whether or not there is an obstacle or not on the image, we delete it from our database.

All images have been passed to our YOLO-based classifier from which we obtained the prediction for the following classes: pedestrians, cars, buses, trucks, animals, and others (referring to unknown obstacles). During this step, the images have been downsized to a dimension of 1920*1080 pixels. As the classifier can detect multiple obstacles in the same image, we only keep the one having the highest probability.
As seen in Fig. 4, for most of the images the neural network is fairly certain to not detect an object but, for a few thousands of pictures, the uncertainty is much more important.

Using a random split strategy, 80% of the dataset has been used to train our algorithms and 20% has been used to test our results. As two succeeding frames of a video are usually visually close and to limit the correlation between the two datasets, the split has been done on parts of the videos (cut into 1s videos) instead of directly on the image.

For simplicity, we choose to limit the weather to only three classes: snow, fog, and "clear" (ie neither snow nor fog). For the same reason, the luminosity conditions are only divided between "night" and "not night".

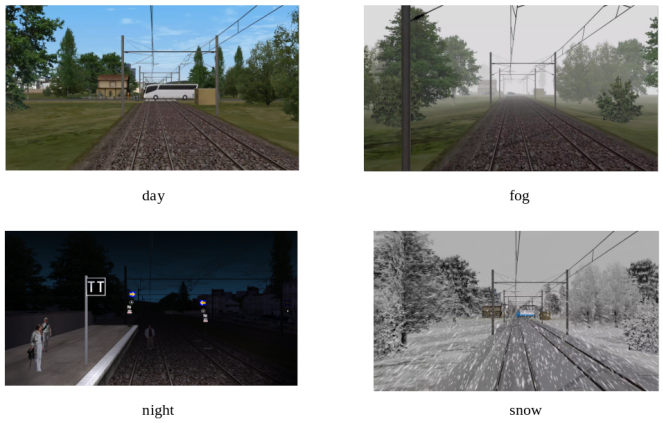

Figure 3. A few examples of the images present in our dataset

C. Safety analysis

In order to demonstrate the possibility to perform a safety analysis of the proposed algorithm, we focused on the construction and analysis of a fault tree. By doing so, we aim to estimate the residual risk associated with the use of the learned tree. In the context of our safety analysis, we identified two undesirable events. The first event is related to the safety and is defined as "the non-application of the emergency braking when an obstacle (human, animal, or object) is present on the track". The second event, linked to the railway performance, is defined as "the application of the emergency braking when no obstacle is present on the track". While the second undesirable event can be triggered by only one false positive, the first undesirable event need to miss the obstacle in every frame of the video.

To demonstrate the possibility to make a fault tree analysis of our algorithm we first need to demonstrate that we can construct the said fault tree and, as we build automatically our fault tree, on a second time that the constructed fault tree is not too complex to be effectively analysed.

1) Decision tree to fault tree: As both of our undesirable events can be decomposed as only two events: "there is no obstacle on the track" (respectively "there is an obstacle on the track") and "the classification tree detect an obstacle" (respectively "the classification tree do not detect an obstacle") and that the second one corresponds to a class output of our decision tree, we choose to automatically extract the set of rules leading to those events from the decision tree. To do so, we used an implementation of the following pseudocode to obtain the set of rules leading to a class (i.e event) :

```
def get_fault_tree(tree, event):
  #return : a list of all the rules leading
  #  to the "event" event.
  #event : a class output of the tree.
  #if there is no such rule "True" will
  # be returned.

    if tree is leaf
      if tree.class == event
            return True
      else:
            return False

    else :
        paths = []

        left_path = get_path(left_child,
                    class)
        right_path = get_path(rigth_child,
                    class)

        if (left_path == True
           & right_path == True)
```

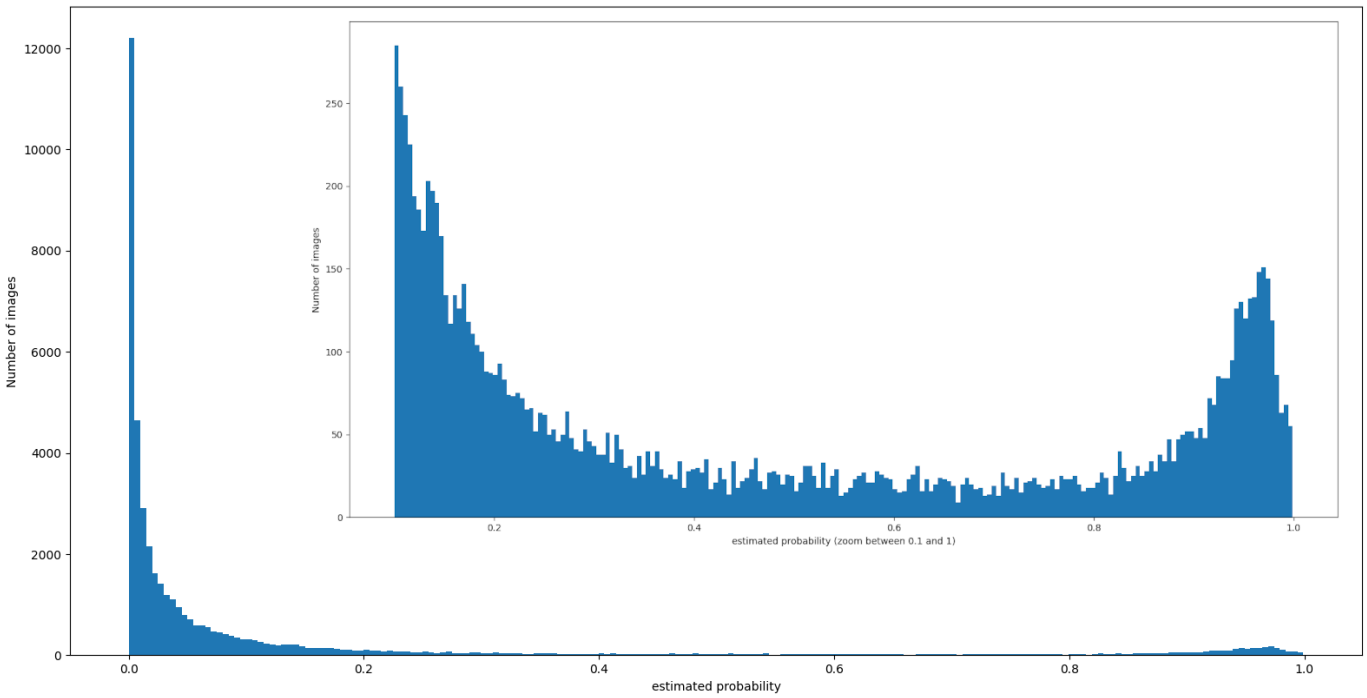

Figure 4. Probability repartition of the predictions made on our dataset using a YOLO-based classifier

```
        return True

    if left_path == True:
      paths = [[["OR", tree.feature,
            "<=", tree.threshold]]]
    else:
        for i in left_path:
          i.append(["AND", tree.feature,
                  "<=" tree.threshold,])
          paths.append(i)

    if right_path == True:
      paths.append([["OR", tree.feature,
                ">", tree.threshold]])
    else:
        for i in right_path:
          i.append(["AND", tree.feature,
                    ">", tree.threshold])
            paths.append(i)

    return paths
```

Coupled with an extra code to delete redundant subrules, the implementation of the aforementioned pseudo-code allows the obtention of the minimal cut of the fault tree. As seen in Fig. 5 and Fig. 6, the obtained set of rules is fairly small facilitating both its analysis and the attribution of basic event probabilities. Note, however, that those sets of rules do not lead to our previously defined undesirable event, but to the events "considering that there is no obstacle" for 6 and "considering that there is an obstacle" for image 5. To obtain our final fault tree as presented in appendix A, we need to incorporate the actual presence or not of an obstacle on the track.

2) Fault tree analysis: The fault trees obtained (see appendix A) from our decision tree are of a size that allows us to effectively analyse them. In order to compute the residual risk induced by our algorithm, we need to make some hypotheses on the probabilities of the presence of the different kinds of obstacles, of the different weather conditions, of the light conditions, and of the repartition of the YOLO-based classifier output. Using information from Meteo France [16] we retained an average of 47 days of fog per year in France. As for the number of days with snow, we retained an average of 20 days in average for France. Finally, we assume that the night covers a third of the travels. We also assume, for simplicity, that train circulation happens uniformly in the time and in the territory. As for the uncertainty to encounter a vehicle (car, bus, or truck) as an obstacle, we retained a probability of $1.5 * 10^{-}3$ per hour. This value corresponds to the statistics of vehicle presence observed on a particular SAL2 level crossing. As the chance to encounter a vehicle on a level crossing is higher than any vehicle outside of a level crossing, we consider this as an acceptable upper bound. As a default value, we used the same value for animals and human's presence on the track.

To fix the probability repartition of the classifier's output (both the class and the estimated probability associated with it) we checked the repartition on our dataset. Such a statistical approach is, to the best of our knowledge, the only feasible way to estimate the output's repartition of a neural network-based classifier. Finally, we also assume that the basic events

```
unknown obstacle AND not fog AND estimated probability <= 0.7522999954223633 AND not day AND not nigth
OR
estimated probability > 0.009900000095367432 AND not fog AND not human AND estimated probability <= 0.7522999954223633 AND not bus AND not day AND not car AND not night
OR
predicted_class = unknown obstacle AND fog AND estimated probability <= 0.7522999954223633
OR
estimated probability > 0.7522999954223633 AND not bus AND not day AND not car AND not night
OR
predicted_class = bus AND not day AND not night
OR
estimated probability > 0.15550000190734863 AND not bus AND day AND not car
OR
estimated probability > 0.60654998779296875 AND fog AND car
OR
bus AND nigth
```

Figure 5. Set of rules leading to consider that there is an obstacle

```
not unknown obstacle AND not fog AND not human AND estimated probability <= 0.7522999954223633 AND not bus AND not clear AND not car AND not night
OR
estimated probability <= 0.3131999969482422 AND estimated probability > 0.009900000095367432 AND not fog AND not human AND not bus AND not clear AND not car AND not nigth
OR
not unknown obstacle AND not truck AND fog AND not human AND estimated probability <= 0.7522999954223633 AND not bus AND not clear AND not car AND not night
OR
truck AND fog AND not human AND estimated probability <= 0.7522999954223633 AND not bus AND not clear AND not car AND not nigth
OR
human AND estimated probability <= 0.7522999954223633 AND not bus AND not day AND not nigth
OR
estimated probability <= 0.15550000190734863 AND not bus AND day AND not car AND not nigth
OR
bus AND day AND not nigth
OR
not fog AND car AND not nigth
OR
estimated probability <= 0.60654998779296875 AND fog AND car AND not nigth
OR
not bus AND nigth
```

Figure 6. Set of rules leading to consider that there is no obstacle

of the fault tree are independent.

Those estimations are used to show the effectiveness of our methodology and might be debatable. However, we believe that they will suffice to demonstrate the interpretability and the possibility to make a safety analysis of the learned decision tree (see appendix B).

A limitation of fault tree's analysis is the necessity to assume independency between events. As such, our probabilities are assumed to be independent during the safety analysis.

IV. Results

As we can see in table III, there is no instance, in the constituted videos dataset, where either the proposed method or the threshold method where unable to detect an obstacle on at least one frame of the video. This means that, at least in our dataset, both the threshold function and the decision tree do not lead to the unwanted event of "the non-application of the emergency braking when an obstacle is present on the track". While only indirectly linked to one of our undesirable events, we can see that the number of false negatives slightly decreases with our method. As the high number of false positives on images is linked to the fact that we have difficulties detecting obstacles at a great distance, we can conclude that we slightly decrease the distance at which we are able to detect an obstacle. The second undesirable event, defined as "the application of the emergency braking when no obstacle is present on the track" being directly linked to the number of false positives, we can see in the table III that our approach reduces by more than 40% the number of undesirable emergency brake use.

Table III
Obtained results with decision tree compared to the results obtained with the threshold

	False positive	False negative (on image)	False negative (on video)	F1-score
Threshold of 50%	222	743	0	0.553
Decision tree	125	714	0	0.599

The calculated probability of occurrence of our undesirable events using the previously described supposition and the constructed fault tree is $5.7669e-006$ for the event of non-application of emergency braking when an obstacle is on the track. While this value is quite high for a safety-related event, please note that not using the emergency brake at all would require not seeing the obstacle on any frame of the video before reaching it and not on only one frame as calculated here. Unfortunately, a fault tree analysis is not able to take into account temporal evolutions and for this reason, we would have to assume independencies between each video frame to estimate the risk of totally missing an obstacle.

Using the previously defined fault tree and the aforementioned assumption, the calculated residual probability of the undesirable event "the application of the emergency braking when no obstacle is present on the track" is 0.031486

While both of those numbers are quite high, they demonstrate that we can analyse, from a safety point of view, the proposed approach.

V. Conclusion

We used an AI approach to learn an interpretable function, reducing the number of undesirable emergency braking of the autonomous train without negative safety impact. We then showed that building a fault tree from our decision function is possible and leads to a fault tree of reasonable size. Finally, using some approximation on the values of the probabilities associated with some events, we calculated the remaining risk of using our learned function in regard to the risk of colliding with vehicles, humans, or animals. While the remaining risk is too important to use our approach as such, we expect that adding more data might increase the performance of the learned function and decrease the remaining risk. Moreover, this work demonstrated the possibility to use a classification tree to increase both the performance and the interpretability of a train's perception system using the weather and the luminosity conditions as complementary information. Finally, while not studied in this work, the proposed methodology could be reused on any system using a deep learning based perception system on changing weather conditions. Depending on the safety guarantee needed for a system wishing to reuse this work, only the decision tree learning and test methodologie might be reused or the safety analysis methodoly using a fault tree might be reused too.

We envision adding more meteorological conditions and the distance between the train and the detected obstacle in the next iteration of our work as we suspect that it would be useful data to enhance our decision tree. Adding more variables to our work will require to use more data and, as the expert's time to annotate them is limited, we plan to use active learning (Aggarwal et al. [17]) in this next iteration of our work. Finally, in an effort to produce a system more resilient we ponder the possibility to replace some of our categorical variables ("fog", "clear", and "night") with numerical data coming from luminosity and humidity sensors.

Acknowledgment

This research work contributes to the french collaborative project TFA (autonomous freight train), with SNCF, Alstom Transport, Hitachi Rail STS, Altran and Apsys. It was carried out in the framework of IRT Railenium, Valenciennes, France, and therefore was granted public funds within the scope of the French Program "Investissements d'Avenir".

References

[1] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788, doi: 10.1109/CVPR.2016.91.

[2] S. Kuutti, S. Fallah, R. Bowden, et P. Barber, ≪ Deep Learning for Autonomous Vehicle Control: Algorithms, State-of-the-Art, and Future Prospects ≫, Synthesis Lectures on Advances in Automotive Technology, vol. 3, no. 4, p. 1-80, août 2019, doi: 10.2200/S00932ED1V01Y201906AAT008.

[3] H. Hadj-Mabrouk Contribution of Artificial Intelligence to Risk Assessment of Railway Accidents. Urban Rail Transit 5, 104–122 (2019). https://doi.org/10.1007/s40864-019-0102-3

[4] H. Hadj-Mabrouk. Contribution of artificial intelligence and machine learning to the assessment of the safety of critical software used in railway transport[J]. AIMS Electronics and Electrical Engineering, 2019, 3(1): 33-70. doi: 10.3934/ElectrEng.2019.1.33

[5] A. Fantini, F. Matteo and M0 Salvatore. (2017). Rock Falls Impacting Railway Tracks: Detection Analysis through an Artificial Intelligence Camera Prototype. Wireless Communications and Mobile Computing. 2017. 1-11. 10.1155/2017/9386928.

[6] J. Xu and B. Ai, "Artificial Intelligence Empowered Power Allocation for Smart Railway," in IEEE Communications Magazine, vol. 59, no. 2, pp. 28-33, February 2021, doi: 10.1109/MCOM.001.2000634.

[7] A. Mahtani, W. Ben-Messaoud, C. Strauss, S. Niar, A. taleb-ahmed. (2020). Pedestrian Detection and Classification for Autonomous Train. 10.1109/IPAS50080.2020.9334938.

[8] NF EN 50128: Railways apn – Communication, signaling and processing systems – Software for railway control and protection systems, 01 October 2011

[9] IEC61508-7. Functional safety of electrical/electronic/programmable electronic safety-related systems, part 7: Overview of techniques and measures. In International Organization for Standardization and International Electrotechnical Commission (2000).

[10] M. Bidoit, F. Losavio, C. Gresse, F. Schlienger. (1986). Automatic programming techniques applied to software development, an approach based on exception handling. Applications of artificial intelligence in engineering problems. Springer, Berlin, Heidelberg, doi: 10.1007/978-3-662-21626-2_15

[11] G.F. Luger, W.A. Stubblefield (1989) Artificial Intelligence and the design of expert systems. Benjamin/Cummings

[12] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone (1983). Classification and Regression Trees.

[13] L. M. Schmidt, G. Kontes, A. Plinge and C. Mutschler, "Can You Trust Your Autonomous Car? Interpretable and Verifiably Safe Reinforcement Learning," 2021 IEEE Intelligent Vehicles Symposium (IV), 2021, pp. 171-178, doi: 10.1109/IV48863.2021.9575328.

[14] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, et al. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825–30.

[15] Hds.utc.fr. 2022. Simulations ferroviaire - UTC Heudiasyc. [online] Available at: ¡https://www.hds.utc.fr/recherche/plateformes-technologiques/simulations-ferroviaire.html¿ [Accessed 13 June 2022].

[16] Meteofrance.com. 2022. Le brouillard — Météo-France. [online] Available at: ¡https://meteofrance.com/comprendre-la-meteo/nuages/le-brouillard¿ [Accessed 13 June 2022].

[17] C. Aggarwal, X. Kong, Q. Gu, J. Han, S.P. Yu. "Active Learning: A Survey." Data Classification: Algorithms and Applications (2014).

[18] C. Guo, G. Pleiss, Y. Sun,K.Q. Weinberger "On Calibration of Modern Neural Networks." Proceedings of the 34th International Conference on Machine Learning, vol.70 (2017), pp.1321-1330

[19] IEC 62290-1 Railway applications: urban guided transport management and command/control systems. Part 1: system principles and fundamental concepts. In International Organization for Standardization and International Electrotechnical Commission (2006).

[20] Hitachi, L., 2022. Heavy Haul Freight Transportation System: AutoHaul : Autonomous Heavy Haul Freight Train Achieved in Australia : Hitachi Review. [online] Hitachi Review. Available at: ¡https://www.hitachi.com/rev/archive/2020/r2020_06/06a05/index.html¿ [Accessed 13 June 2022].

[21] DIGITALSNCF. 2022. Train Autonome Voyageurs et Fret : deux projets désormais en route. [online] Available at: ¡https://www.digital.sncf.com/actualites/train-autonome-voyageurs-et-fret-deux-projets-desormais-en-route¿ [Accessed 13 June 2022].

[22] J. Huang, Y. Liu, Y. Xia, Z. Zhong, and J. Sun, "Train Driving Data Learning with S-CNN Model for Gear Prediction and Optimal Driving," in 2019 Chinese Automation Congress. IEEE, 2019, pp. 2227–2232.

[23] J. Yin, D. Chen and L. Li, "Intelligent Train Operation Algorithms for Subway by Expert System and Reinforcement Learning," in IEEE Transactions on Intelligent Transportation Systems, vol. 15, no. 6, pp. 2561-2571, Dec. 2014, doi: 10.1109/TITS.2014.2320757.

[24] J. Huang, F. Yang, Y. Deng, X. Zhao and M. Gu, "Human experience knowledge induction based intelligent train driving," 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 335-340, doi: 10.1109/ICIS.2017.7960015.

[25] J. Huang, Y. Cai, J. Li, X. Chen and J. Fan, "Toward Intelligent Train Driving through Learning Human Experience," 2019 1st International

Conference on Industrial Artificial Intelligence (IAI), 2019, pp. 1-6, doi: 10.1109/ICIAI.2019.8850749.

APPENDIX

A. Fault tree

Figure 7. Fault tree of the event ”non application of the emergency braking when an obstacle is present on the track”. (each gamma is a probability)

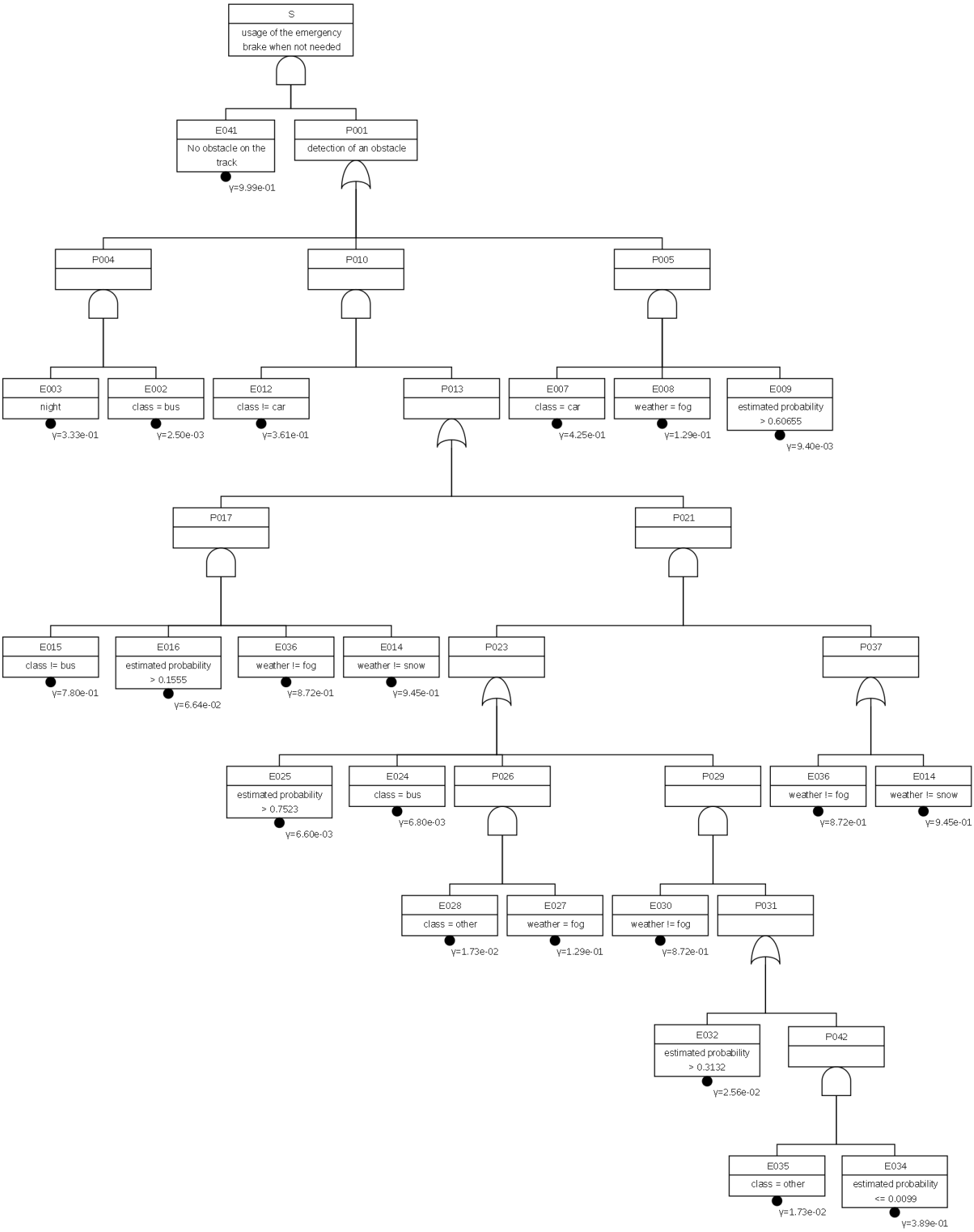

Figure 8. Fault tree of the event ”application of the emergency braking when no obstacle is present on the track”. (each gamma is a probability)

B. Learned decision tree

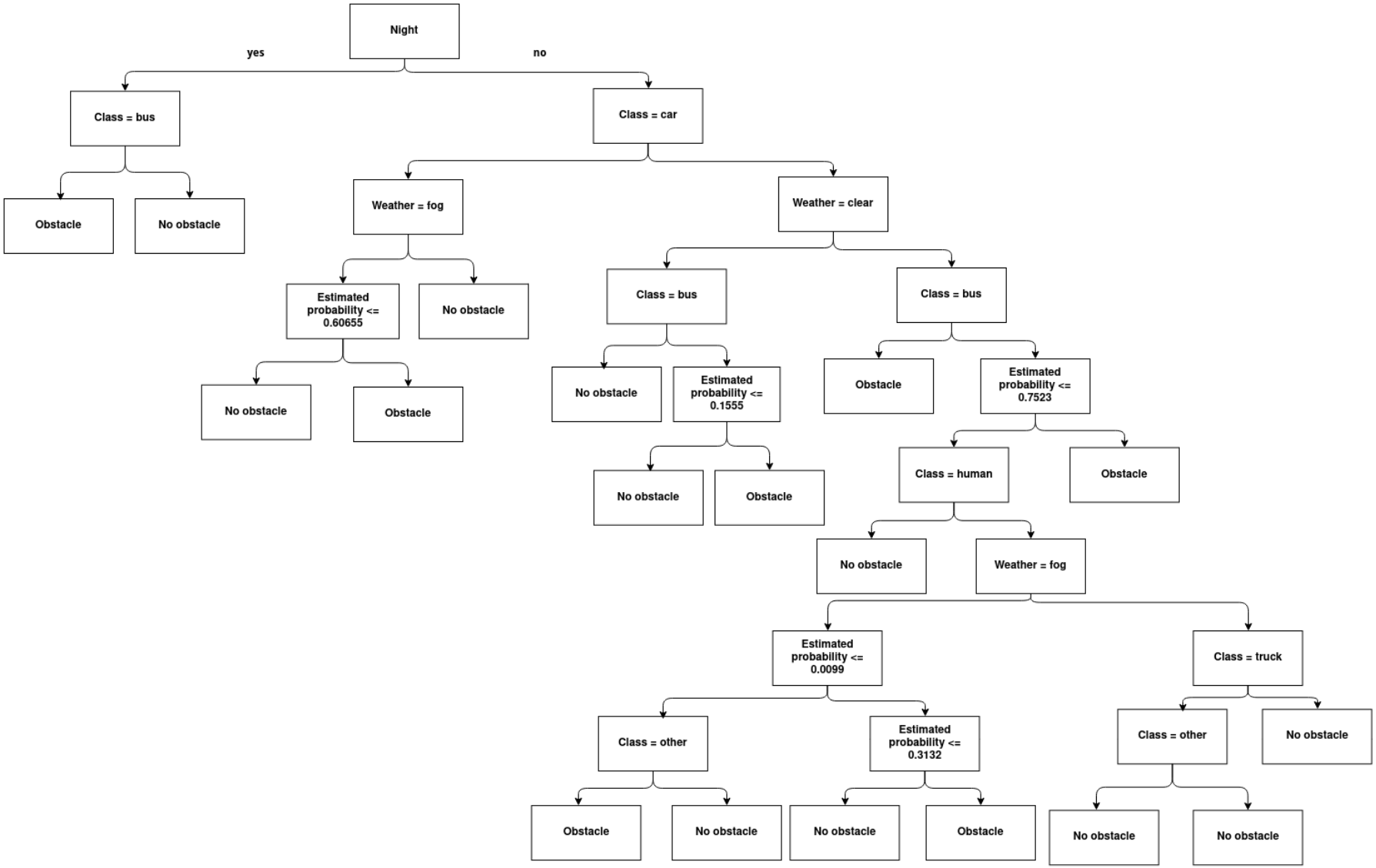

Figure 9. Learned decision tree