Closed-Loop Analysis of Vision-Based Autonomous Systems: A Case Study

Corina S. Păsăreanu[1,2(✉)], Ravi Mangal[2], Divya Gopinath[1], Sinem Getir Yaman[3], Calum Imrie[3], Radu Calinescu[3], and Huafeng Yu[4]

[1] KBR, NASA Ames, Moffett Field, CA 94035, USA
pcorina@andrew.cmu.edu
[2] Carnegie Mellon University, Moffett Field, CA 94035, USA
[3] University of York, York, UK
[4] Boeing Research and Technology, Santa Clara, CA, USA

Abstract. Deep neural networks (DNNs) are increasingly used in safety-critical autonomous systems as perception components processing high-dimensional image data. Formal analysis of these systems is particularly challenging due to the complexity of the perception DNNs, the sensors (cameras), and the environment conditions. We present a case study applying formal probabilistic analysis techniques to an experimental autonomous system that guides airplanes on taxiways using a perception DNN. We address the above challenges by replacing the camera and the network with a compact abstraction whose transition probabilities are computed from the confusion matrices measuring the performance of the DNN on a representative image data set. As the probabilities are estimated based on empirical data, and thus are subject to error, we also compute confidence intervals in addition to point estimates for these probabilities and thereby strengthen the soundness of the analysis. We also show how to leverage local, DNN-specific analyses as run-time guards to filter out mis-behaving inputs and increase the safety of the overall system. Our findings are applicable to other autonomous systems that use complex DNNs for perception.

1 Introduction

Complex autonomous systems, such as autonomous aircraft taxiing systems [31] and autonomous cars [20,25,42], need to perceive and reason about their environments using high-dimensional data streams (such as images) generated by rich sensors (such as cameras). Machine learnt components, specially deep neural networks (DNNs), are particularly capable of the required high-dimensional reasoning and hence, are increasingly used for perception in these systems. While formal analysis of the safety of these systems is highly desirable due to their safety-critical operational settings and the error-prone nature of learned components, in practice this is very challenging because of the complexity of the system components, including the high complexity of the neural networks (which may have thousands or millions of parameters), the complexity of the camera capture

C. Enea and A. Lal (Eds.): CAV 2023, LNCS 13964, pp. 289–303, 2023.
https://doi.org/10.1007/978-3-031-37706-8_15

process, and the random and hard to characterize nature of the environment in which the system operates (i.e., the world itself).

In this work, we describe a formal analysis of a closed-loop autonomous system that addresses the above challenges. Our case study is motivated by a real-world application, namely, an experimental autonomous system for guiding airplanes on taxiways developed by Boeing [3,14]. The key idea is to abstract away altogether the perception components, namely, the perception network and the image generator, i.e., the camera taking images of the world, and replace them with a probabilistic component α that maps (abstractions of) the state of the system to state estimates that are used in downstream decision making in the closed-loop system. The resulting system can then be analyzed with standard (probabilistic) model checkers, such as PRISM [34] or STORM [22].

The approach is *compositional*, in the sense that the probabilistic component is computed separately from the rest of the system. The transition probabilities in α are derived based on *confusion matrices* computed for the DNN (measured on representative data sets). Developers routinely use confusion matrices to evaluate machine learning models, so our analysis is closely aligned with existing workflows, facilitating its adoption in practice.

The size of the probabilistic abstraction α is linear in the size of the output of the DNN, and is independent of the number of the DNN parameters or the complexity of the camera and the environment. We also describe how to leverage additional results obtained from analyzing the DNN in isolation to further refine the abstraction and also increase the safety of the closed-loop system through *run-time guards*. In particular, we leverage rules mined from the DNN model [17] to act as run-time guards for the closed-loop analysis, filtering out inputs that likely lead to invalid DNN behavior. Other methods can also be used (e.g. [17, 18,21,26,32,35]) to catch adversarial or out-of-distribution inputs.

The probabilities in α are estimated based on empirical data, so they are subject to error. We explore the use of *confidence intervals* in addition to point estimates for these probabilities and thereby strengthen the soundness of the analysis [5,7]. Our technique is applicable to other autonomous systems that use DNN-based perception from high-dimensional data.

Related Work. Formal proofs of closed-loop safety have been obtained for systems with low-dimensional sensor readings [11,12,27–30,40]; however, they become intractable for systems that use rich sensors producing high-dimensional inputs such as images.

Other works address the modeling and scalability challenges by constructing *abstractions* of the perception components [24,33]. To model different environment conditions, these abstract models use *non-deterministic* transitions. The resulting closed-loop systems are analyzed with traditional (non-probabilistic) techniques. The abstractions either lack soundness proofs [33] or come with only probabilistic soundness guarantees [24] which do not translate into probabilistic guarantees over the safety of the overall system. VerifAI [16] can find counter-examples to system safety, but can not provide guarantees.

The recent work in [36] aims to verify the safety of the trajectories of a camera-based autonomous vehicle in a given 3D-scene. The work use invariant

regions over the input space grouped based on the same controller action. However, their abstraction captures only one environment condition (i.e., one scene) and one camera model, whereas our approach is not particular to a camera model and implicitly considers all the possible environment conditions.

In contrast to previous work, we describe a formal analysis that is *probabilistic*, which we believe is natural since the camera images capturing the state of the world are subject to randomness due to the environment; further DNNs are learnt from data and are not guaranteed to be 100% accurate. Recent work [2] also discusses the use of classification metrics, such as confusion matrices, for quantitative system-level analysis with temporal logic specifications. However, the work does not discuss the computation of confidence intervals that is necessary for quantifying the empirical results. Also, it does not incorporate DNN specific analyses as we do here. We build on our previous work DeepDECS [6], where the goal is to perform controller synthesis with safety guarantees, so the formalism is more involved. Furthermore, DeepDECS does not consider confidence interval analysis, which we explore here based on some of our other previous works [5,7]. We analyzed center-line tracking using TaxiNet in [31]. That work focuses on the analysis of the network and not on the overall system.

2 Autonomous Center-Line Tracking with TaxiNet

Boeing is developing an experimental autonomous system for center-line tracking on taxiways in an airport. The system uses a neural network called TaxiNet for perception. TaxiNet is designed to take a picture of the taxiway as input and return the plane's position with respect to the center-line on the taxiway. It returns two outputs; cross track error (`cte`), which is the distance in meters of the plane from the center-line and heading error (`he`), which is the angle in degrees of the plane with respect to the center-line. These outputs are fed to a controller which in turn manoeuvres the plane such that it remains close to the center of the taxiway. This forms a closed-loop system where the perception network continuously receives images as the plane moves on the taxiway. We use this system as a case study and also as a running example throughout the paper.

System Decomposition. The decomposition of this system is illustrated in Fig. 1. The controller sends actions a to the airplane to guide it on the taxiway. The dynamics (which models the movement of the airplane on the airport surface) maps previous state s and action a to the next state s'.[1] Information about the taxiway is provided by the perception network (p), i.e. TaxiNet. The perception network takes high-dimensional images captured with a *camera* (c), and returns its estimation s_{est} of the real state s.

For our application, state $s \in S$ captures the position of the airplane on the surface; S is modeled as $\texttt{CTE} \times \texttt{HE}$. The network estimates the state $s := (\texttt{cte}, \texttt{he})$ based on images taken with a camera placed on the airplane. If the network is 'perfect', then $s = s_{est}$.[2] However, this does not hold in practice.

[1] Velocity may be provided as feedback to the controller; we ignore here for simplicity.
[2] Assuming the relevant state of the system is recoverable from the input image.

The network is trained on a finite set of images and is not guaranteed to be 100% accurate whereas images observed in operation show a wide variety due to different environment (e.g., light, weather) conditions and imperfections in the camera.

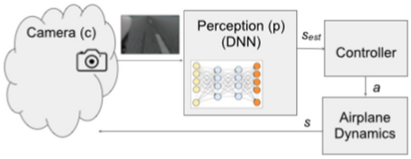


Fig. 1. Closed-loop System

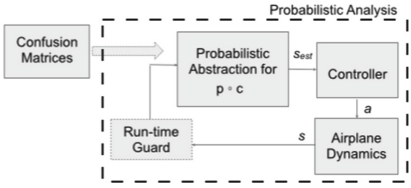


Fig. 2. Abstracted System

Component Modeling. We built a simple discrete model of the airplane dynamics and a discrete-time controller for the system, similar to previous related work [4,23] which also considers discretized control. Since the controller is discretized, we abstract the regression outputs of TaxiNet to view the model as a classifier which predicts the plane's position in discrete states. Treatment of more complex systems with continuous semantics and regression models is left for future work. The main challenge that we address in the paper is the modeling of the perception components (the camera capture process and the network), which we describe in detail in the next section. We model the (abstracted) autonomous system as a Discrete Time Markov Chain (DTMC) [38]; the code for the models is provided in the appendix of an extended version of this paper [37].

Safety Properties. In our study, the goal is to provide *guarantees* for safe behavior with respect to two system-level properties indicated by our industrial partner. The properties specify conditions for safe operation in terms of allowed `cte` and `he` values for the airplane, by using taxiway dimensions. The first property states that the airplane shall never leave the taxiway (i.e., $|\texttt{cte}| \leq 8$ meters). The second property states that the airplane shall never turn more than a prescribed degree (i.e., $|\texttt{he}| \leq 35°$), as it would be difficult to maneuver the airplane from that position. These two properties can be encoded in PCTL [8] as follows.

$$P =?[F(|\texttt{cte}| > 8\,m)] \qquad (\textit{Property 1})$$

$$P =?[F(|\texttt{he}| > 35°)] \qquad (\textit{Property 2})$$

Here $P =?$ indicates that we want to calculate the probability that eventually (F) the system reaches an error state.

TaxiNet DNN. This is a regression model with 24 layers including five convolution layers, and three dense layers (with 100/50/10 ELU neurons) before the output layer. The inputs to the model are RGB color images of size 360 × 200 pixels. We use a representative data set with 11108 images, shared by our industry partner. The model has a Mean Absolute Error (MAE) of 1.185 for

cte and 7.86 for he outputs respectively. The discrete nature of the controller in our DTMCs induces a discretization on TaxiNet's outputs and the treatment of TaxiNet as a classifier for the purpose of our analysis. $\texttt{cte} \in [-8.0\text{ m}, 8.0\text{ m}]$ and $\texttt{he} \in [-35.0°, 35.0°]$ are translated into $\underline{\texttt{cte}} \in \{0, 1, 2, 3, 4\}$ and $\underline{\texttt{he}} \in \{0, 1, 2\}$ as shown below.

$$\underline{\texttt{cte}} = \begin{cases} 3 \text{ if } -8.0\text{ m} <= \texttt{cte} < -4.8\text{ m} \\ 1 \text{ if } -4.8\text{ m} <= \texttt{cte} < -1.6\text{ m} \\ 0 \text{ if } -1.6\text{ m} <= \texttt{cte} <= 1.6\text{ m} \\ 2 \text{ if } 1.6\text{ m} < \texttt{cte} <= 4.8\text{ m} \\ 4 \text{ if } 4.8\text{ m} < \texttt{cte} <= 8.0\text{ m} \end{cases} \qquad \underline{\texttt{he}} = \begin{cases} 1 \text{ if } -35.0° <= \texttt{he} < -11.67° \\ 0 \text{ if } -11.67° <= \texttt{he} <= 11.66° \\ 2 \text{ if } 11.66° < \texttt{he} <= 35.0° \end{cases}$$

We use label "−1" to denote error states, i.e., $\underline{\texttt{cte}} = -1$ iff $|\texttt{cte}| > 8$ m and $\underline{\texttt{he}} = -1$ iff $|\texttt{he}| > 35°$. For simplicity, we use cte and he to denote both the classifier and regression outputs in other parts of the paper (with meaning clear from context). Note that none of the input images are labeled by the classifier as "−1", as the outputs of the network are normalized to be within the prescribed bounds; however, this does not preclude the system from reaching an error.

3 Probabilistic Analysis

In this section, we describe the methodology for abstracting and analyzing an autonomous system leveraging probabilistic model checking. The main idea, which we initially explored in [6], is to replace the composition $p \circ c$ of the camera (denoted as c) and the perception DNN (denoted as p) with a conservative abstraction mapping each system state to every possible estimated state; the transition probabilities are derived empirically based on the confusion matrices computed for the DNN, on a representative data set. We denote this abstraction as $\alpha : S \to \mathcal{D}(S)$, mapping system states to a discrete distribution over (estimated) system states. Figure 2 depicts the abstracted autonomous system.

We observe that c can be viewed as a map between state $s \in S$ to a distribution over images, denoted as $\mathcal{D}(\texttt{Img})$, where $\texttt{img} \in \texttt{Img}$ and Img is the set of images. For instance, in the TaxiNet system, state s only captures the position of the airplane with respect to the center-line, but there are many different images that correspond to the same position. This is due to uncontrollable environmental conditions, such as temporary sensor failures or different lighting and weather conditions. Consequently, a single state s can map to a number of different images depending on the environment, and this is modeled by considering c to be a probabilistic map of type $S \to \mathcal{D}(\texttt{Img})$. Given a system state s, $\alpha(s)$ models the probability of $p \circ c$ leading to a particular estimated state s_{est}; α needs to be probabilistic because c itself is probabilistic and p is not perfectly accurate.

We further describe how we can leverage DNN-specific analysis to improve the accuracy of perception and the safety of the overall system, via the optional addition of run-time guards. For the verification of the closed-loop system, we use the PRISM model checking tool [34]. We also explore methods for analysis

of DTMCs with uncertain transition probabilities [5,7], to obtain *probabilistic guarantees* about the validity of our probabilistic safety proofs even though the abstraction probabilities are empirical estimates.

Assumptions. Our analysis assumes that the distribution of inputs to the network remains fixed over time (i.e., it is not subject to distribution shifts). Moreover, the data set of input images used to estimate the probabilities in α is assumed to be *representative*, i.e., constituted of independently drawn samples from this fixed underlying distribution of inputs. Relaxing these assumptions is a challenging but important task for future research.

3.1 Probabilistic Abstractions for Perception

We describe in detail the construction of the probabilistic abstraction $\alpha : S \to \mathcal{D}(S)$. We do not need access to the camera and only require black-box access to the network for constructing our abstraction.[3] We assume S is a finite set such that $\#S = K$ where $\#S$ denotes the cardinality of set S. We use $\alpha(s, s_{est})$ to represent the probability associated with estimated state s_{est}. It is defined as,

$$\alpha(s, s_{est}) := \Pr_{\texttt{img} \sim c(s)}[p(\texttt{img}) = s_{est}] \tag{1}$$

We estimate the probabilities in α by means of a confusion matrix. Let $\overline{\texttt{Img}}_s \subseteq \texttt{Img}$ denote a *representative test dataset* for images corresponding to state s, i.e., every sample in $\overline{\texttt{Img}}_s$ is assumed to be an independently drawn sample from $c(s)$. We assume access to representative test datasets corresponding to every state $s \in S$. Let $\overline{\texttt{Img}} := \bigcup_{s \in S} \overline{\texttt{Img}}_s$. For any test input $\texttt{img} \in \overline{\texttt{Img}}$, let $p^*(\texttt{img}) \in S$ be the label (i.e., the true underlying state) of $\texttt{img}$, which is known since $\overline{\texttt{Img}}$ is a test dataset. For the sake of technical presentation, we assume a bijective map $\texttt{rep} : S \to [K]$ that maps every state in S to a number in $[K] := \{1, 2, \ldots, K\}$. We evaluate p on the test dataset $\overline{\texttt{Img}}$ to construct a $K \times K$ confusion matrix $\mathcal{C}$ such that, for any $k, k' \in [K]$, the element in row k and column k' of this matrix is given by the number of inputs from $\overline{\texttt{Img}}$ with true state $\texttt{rep}^{-1}(k)$ that the perception network p classifies as state $\texttt{rep}^{-1}(k')$.

$$\mathcal{C}[k, k'] := \#\left\{\texttt{img} \in \overline{\texttt{Img}} \mid p^*(\texttt{img}) = \texttt{rep}^{-1}(k) \wedge p(\texttt{img}) = \texttt{rep}^{-1}(k')\right\} \tag{2}$$

Given the confusion matrix $\mathcal{C}$, empirical estimates for the probabilities in α are calculated as follows,

$$\alpha(\texttt{rep}^{-1}(k), \texttt{rep}^{-1}(k')) := \frac{\mathcal{C}[k, k']}{\sum_{k'' \in [K]} \mathcal{C}[k, k'']}. \tag{3}$$

[3] Our run-time guard does require white-box access.

TaxiNet Example. For the TaxiNet application, we construct two probabilistic maps, $\alpha_{\texttt{cte}}$ and $\alpha_{\texttt{he}}$, corresponding to each of the state variables `cte` and `he`, using a representative test data set with 11108 samples.[4] Thus, $\alpha_{\texttt{cte}}$ is of type $\texttt{CTE} \to \mathcal{D}(\texttt{CTE})$ and $\alpha_{\texttt{he}}$ is of type $\texttt{HE} \to \mathcal{D}(\texttt{HE})$. Table 1 illustrates the confusion matrix for `he`. The mapping $\alpha_{\texttt{he}}$ is computed in a straightforward way: $\alpha_{\texttt{he}}(0,0) = 4748/(4748 + 2139 + 148) = 0.675$, giving the probability of estimating correctly that the value of `he` is zero. Similarly, $\alpha_{\texttt{he}}(1,0) = 91/(91 + 2010) = 0.043$, giving the probability of estimating incorrectly that the value of `he` is zero instead of one. The corresponding DTMC code is as follows:

		Predicted		
	Total = 11108	**0**	**1**	**2**
Actual	**0**	4748	2139	148
	1	91	2010	0
	2	744	211	1017

Table 1. Confusion Matrix for `he`

```
[] he=0 → 0.675: (he_est'=0) + 0.304: (he_est'=1) + 0.021: (he_est'=2);
[] he=1 → 0.043: (he_est'=0) + 0.957: (he_est'=1) + 0.0: (he_est'=2);
[] he=2 → 0.377: (he_est'=0) + 0.107: (he_est'=1) + 0.516: (he_est'=2);
```

A similar computation is performed for constructing $\alpha_{\texttt{cte}}$. The resulting code for the closed-loop system is shown in [37], in the appendix.

3.2 DNN Checks as Run-Time Guards

We use DNN-specific checks as run-time guards to improve the performance of the perception network and therefore the safety of the overall system. We hypothesize that for inputs where the checks pass, the network is more likely to be accurate, and therefore, the system is safer.

For our case study, we distill logical rules from the DNN that characterize misbehavior in terms of intermediate neuron values and use them as run-time guards (as described in Sect. 4). More generally, one can use any off-the-shelf pointwise DNN check, such as local robustness [10,15,19,35,39,41] or confidence checks for well-calibrated networks [21], as run-time guards (provided that they are fast enough to be deployed in practice). For practical reasons (TaxiNet is a regression model, it contains ELU [9] activations, we do not have access to the training data) we can not use off-the-shelf checks here.

Modeling DNN Checks. Let us denote the application of (one or more) DNN-specific checks as a function $\texttt{check} : (\texttt{Img} \to S) \times \texttt{Img} \to \mathbb{B}$, such that, for perception network $p \in \texttt{Img} \to S$ and image $\texttt{img} \in \texttt{Img}$, $\texttt{check}(p, \texttt{img}) = \texttt{true}$ if p passes the checks at input `img`, and $\texttt{check}(p, \texttt{img}) = \texttt{false}$ otherwise.

We further assume that a system that uses DNN checks as a run-time guard attempts to read the camera sensor multiple (one or more) times, until the check passes; and aborts (or goes to a fail-safe state) if the number of consecutive failed checks reaches a certain threshold. This logic can be generalized to consider more sophisticated safe-mode operations; for instance, the system can decelerate

[4] To simplify the DTMCs, we model the updates to `cte` and `he` as independent. For more precision, we can compute confusion matrices and α for the pair (`cte`, `he`).

and/or notify an operator when the threshold is reached, as this could indicate serious sensor failure or adverse weather conditions.

To model the effect of the run-time check in our analysis, we can define β as the probability that an image `img` generated by the camera c, for *any* state s, satisfies $\texttt{check}(p, \texttt{img}) = \texttt{true}$;

$$\beta := \Pr_{\texttt{img} \sim D}[\texttt{check}(p, \texttt{img}) = \texttt{true}] \tag{4}$$

Here D is the distribution obtained by *combining* $c(s)$ for all states $s \in S$.[5] To be more precise we can define a separate β_s for each state s. We estimate β using the representative set of images $\overline{\texttt{Img}}$,

$$\beta := \frac{\#\overline{\texttt{Img}}^{true}}{\#\overline{\texttt{Img}}} \tag{5}$$

where $\overline{\texttt{Img}}^{true} := \{\texttt{img} \in \overline{\texttt{Img}} \mid \texttt{check}(p, \texttt{img}) = \texttt{true}\}$.

For the overall analysis of the closed-loop system, irrespective of the state s, we can assume that the DNN check will pass with a probability β. Moreover, since the perception network only processes images that pass the DNN check, we construct a refined probabilistic abstraction α^{true} using conditional probability:

$$\alpha^{true}(s, s_{est}) := \Pr_{\texttt{img} \sim c(s)}[p(\texttt{img}) = s_{est} | \texttt{check}(p, \texttt{img}) = \texttt{true}] \tag{6}$$

We can estimate α^{true} as before, but the confusion matrix is built using only the images that pass the DNN check, i.e., for dataset $\overline{\texttt{Img}}^{true} \subseteq \overline{\texttt{Img}}$.

TaxiNet Example. For TaxiNet, out of 11108 inputs, 9125 inputs (i.e., 82.1%) pass the DNN check resulting in the following code:

```
i:[0..M] init 0;
[]  pc=0 & i<M → 0.821: (v'=1) & (pc'=1) & (i'=0) + 0.179: (v'=0) & (i'=i+1);
```

We model the result of applying the DNN check with variable v; $v = 1$ if the check returns `true` for an image and $v = 0$ otherwise. M is the number of allowed repeated sensor readings and i is used to count the number of failed DNN checks.

The abstraction for state variables `he` ($\alpha_{\texttt{he}}$) and `cte` ($\alpha_{\texttt{cte}}$) is only computed for the inputs that pass the check (i.e., for $v = 1$) based on newly computed confusion matrices. The DTMC code for the closed-loop system with run-time guards is shown in [37], in the appendix.

3.3 Confidence Analysis

The construction of the probabilistic abstractions relies on calculating empirical point estimates of the required probabilities. However, these empirical estimates lack statistical guarantees and can be off by an arbitrary amount from the true probabilities. To address this concern, we experiment with using FACT [5,7]

[5] To simplify the presentation, we omit the precise mathematical formulation for D.

to calculate *confidence intervals* for the probability that the safety properties of the closed-loop system are satisfied. The inputs to FACT are: 1) a parametric DTMC m where each empirically estimated transition probability is represented by a parameter, 2) a PCTL formula ϕ, 3) an error level $\delta \in (0, 1)$ and 4) an *observation function* O mapping state s to a tuple representing the number of observations for each outgoing transition from s; in our case, the number of observations can be obtained directly from the computed confusion matrices, i.e., $O(s) = (\mathcal{C}[\texttt{rep}(s), 1], \ldots, \mathcal{C}[\texttt{rep}(s), K])$. FACT synthesizes a $(1 - \delta)$-confidence interval $[a, b] \subseteq [0, 1]$ for the probability that ϕ is satisfied, given the observations.

TaxiNet Example. The following partial code illustrates the parametric version of the code provided in Sect. 3.1 (with the complete code for the parametric models provided in [37], in the appendix). The first three lines represent the number of observations obtained from the confusion matrix in Table 1.

```
param double x = 4748 2139 148;
param double y = 91 2010;
param double z = 744 211 1017;
...
[] he=0 → x1:(he_est'=0) + x2:(he_est'=1) + (1-x1-x2):(he_est'=2);
[] he=1 → y1:(he_est'=0) + (1-y1):(he_est'=1);
[] he=2 → z1:(he_est'=0) + z2:(he_est'=1) + (1-z1-z2):(he_est'=2);
```

4 Experiments

In this section, we report on the experiments that we conducted as part of our probabilistic safety analysis of the center-line tracking autonomous system.

We built two DTMC models, m_1 and m_2, denoting the closed-loop center-line tracking system without and with a run-time guard, respectively. The airplane dynamics and the controller are identically modeled in the two DTMCs as discrete components. The code for the models (in PRISM syntax) and more details about the analysis are presented in [37], in the appendix.

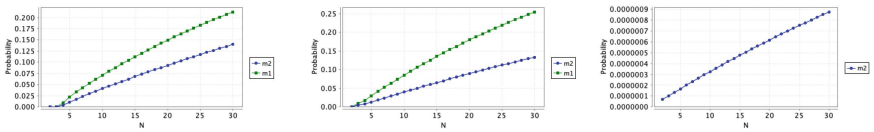
Mining Rules for Run-time Guards. We leverage our prior work [17], to extract rules of the form $Pre \implies Post$ from the DNN. $Post$ is the condition $|\texttt{cte}^* - \texttt{cte}| > 1.0$ m $\vee\ |\texttt{he}^* - \texttt{he}| > 5°$ on the regression model's outputs and Pre is a condition over the neuron values in the three dense layers of TaxiNet ($\texttt{cte}^*$ and $\texttt{he}^*$ denote ground-truth values). The considered $Post$ characterizes invalid behavior (as explained in [31]). If an input satisfies Pre, the DNN check is considered to have failed on that input. Pre can be evaluated efficiently during the forward pass of the model, making it a good run-time guard candidate. Here is an example of a rule for invalid behavior:

$$\begin{aligned} N_{1,85} <= -0.998 \ \wedge\ N_{2,50} <= 3.31 \ \wedge\ N_{1,84} <= -0.994 \ \wedge\ N_{1,15} > -0.999 \\ \wedge\ N_{1,21} <= 1.711 \ \wedge\ N_{1,70} <= 11.088 \ \wedge\ N_{1,51} > -0.999 \ \wedge\ N_{1,21} > -0.637 \implies \\ |\texttt{cte}^* - \texttt{cte}| > 1.0\ m \ \vee\ |\texttt{he}^* - \texttt{he}| > 5° \end{aligned}$$

$N_{i,j}$ indicates the j^{th} neuron in the i^{th} dense layer. The conditions over neuron values can be checked during the forward pass of the DNN. If an input satisfies the conditions, it is interpreted as failing the check. If the check consecutively

fails M times, the system aborts, meaning that the system stops operating and hands over control to a fail-safe mechanism (such as the pilot). More details on the rules and their deployment as run-time guards are in [37], in the appendix.

Confusion Matrices. The confusion matrices for the classification version of TaxiNet, computed for the two cases (without and with run-time guard) are shown in [37], in the appendix. The tables can be used by developers to better understand the DNN performance. For instance, the results summarized in the confusion matrices indicate that the DNN performs best for inputs lying on the center-line, which can be attributed to training being done mainly using scenarios where the plane follows the center-line. The model appears to perform better when the plane is heading left, as opposed to heading right, which may be due to camera position. These observations can be used by developers to improve the model, by training on more scenarios. Note also that the model does not make 'blatant' errors, mistaking inputs on the *left* as being on the *right* (of center-line) or vice-versa (see e.g., entries with zero observations). Formal proofs can provide guarantees of absence of such transitions.



(a) *Property* 1 (b) *Property* 2 (c) *Property* 3

Fig. 3. Probabilistic model checking results via PRISM

Analysis. We analyzed m_1 and m_2 with respect to the two PCTL properties, $P =?[F(\texttt{cte} = -1)]$ (*Property 1*), and $P =?[F(\texttt{he} = -1)]$ (*Property 2*)[6]. The airplane is assumed to start from a initial position on the center-line and heading straight. For m_2, i.e. the model with a run-time guard, we also evaluate the probability of the TaxiNet system going to the abort state using the property, $P =?[F(v = 0 \ \& \ i = M)$ (*Property 3*), where M is the threshold for the number of consecutive run-time check failures.

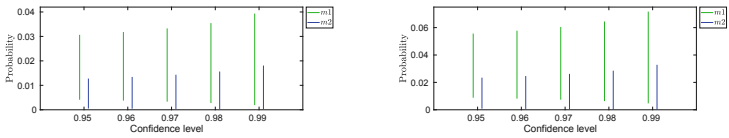
The probabilities of these properties being satisfied, calculated by PRISM, are shown in Fig. 3, where N is a constant in the DTMCs that dictates the length of the finite-time horizon considered for the analysis. Note that the system has an additional planning layer that calculates the waypoints for the airplane's course on the taxiway. The system is only used for controlling the airplane movement between pairs of waypoints, hence a short horizon suffices.

The confidence intervals computed with FACT are shown in Fig. 4, at different confidence levels (0.95 to 0.99), for $N = 4$. For computing the intervals, we ignore the transitions in the DTMCs that were not observed in our data (see [37] for more details).

[6] We rewrote the properties in terms of the discrete values.

The PRISM analysis scales well; e.g., evaluating *Property 1* for model m_2 ($N = 30$) requires less than 0.1 s on an M1 MacBook Pro, 16 GB RAM. The numbers are similar for other queries. However, the confidence analysis does not scale as well; we could not go beyond $N = 4$ for a timeout of two hours, with *Property 1* hardest to check. Newer work, fPMC [13], addresses these scalability challenges but we found it not yet mature enough to be applied to our models.

Discussion and Lessons Learned. The experiments demonstrate the feasibility of our approach, which enables reasoning about a complex DNN interacting with conventional (discrete-time) components via a simple probabilistic abstraction. Our analysis not only provides qualitative (i.e., an error is reachable or not) but also quantitative (i.e., likelihood of error) results, helping developers assess the risk associated with the analyzed scenario.



(a) *Property* 1 (b) *Property* 2

Fig. 4. Confidence interval results via FACT

The results highlight the benefit of the run-time guards in improving the safety of the overall system; see Figs. 3(a,b) for lower error probabilities and Figs. 4(a,b) for tighter intervals for m_2. The probability of aborting is very small, indicating the efficacy of the fail-safe mechanism (see Figs. 3(c)). More importantly, since the DNN demonstrates higher accuracy on the inputs where the run-time check passes, the results also indicate that *improved accuracy of the DNN translates into improved safety*. The computed probabilities and confidence intervals can be examined by developers and regulators to ensure that system safety is met at required levels. If the confidence intervals are too large, they can be made tighter by adding more data, as guided by the confusion matrices.

Based on our feedback (confusion matrices) our industrial partner is retraining the perception network. As the system is in its early stages, our industrial partner was more interested in the trends suggested by our analysis rather than the exact probability results. For instance, our results indicate that safety will increase with a better-performing network. The partner was also interested in how the DNN-specific analysis contributes to the system-level analysis. A probabilistic analysis is best viewed as an "average-case" analysis rather than "worst-case". Nevertheless, such analysis is still useful since it conveys whether the system at least behaves safely in the average-case.

5 Conclusion

We demonstrated a method for the analysis of the safety of autonomous systems that use complex DNNs for visual perception. Our abstraction helps separate the concerns of DNN and conventional system development and evaluation. It also enables the integration of heterogeneous artifacts from DNN-specific analysis and system-level probabilistic model checking. The approach produces not only qualitative results but also provides insights that can be used in quantitative safety assessment for AI/DNN-enabled systems. This is, potentially, an important step to fill one of the gaps of quantitative evaluation for future AI certification [1].

Future work involves experimentation with image data sets representing a variety of environment conditions. We also plan to refine our models, inducing finer partitions on the DNN, and validate them through simulations. Another future research direction involves the study of the composition of safety proofs for the system analyzed in different scenarios. Finally, we are working on compositional analysis techniques to achieve worst-case (non-probabilistic) guarantees.

References

1. EASA concept paper: First usable guidance for level 1 machine learning applications (2021). https://www.easa.europa.eu/en/downloads/134357/en
2. Badithela, A., Wongpiromsarn, T., Murray, R.M.: Leveraging classification metrics for quantitative system-level analysis with temporal logic specifications. In: 2021 60th IEEE Conference on Decision and Control (CDC), pp. 564–571. IEEE (2021). https://doi.org/10.1109/CDC45484.2021.9683611
3. Beland, S., et al.: Towards assurance evaluation of autonomous systems. In: IEEE/ACM International Conference On Computer Aided Design, ICCAD 2020, San Diego, CA, USA, 2–5 November 2020, pp. 84:1–84:6. IEEE (2020)
4. Byrne, R., Abdallah, C., Dorato, P.: Experimental results in robust lateral control of highway vehicles. In: Proceedings of 1995 34th IEEE Conference on Decision and Control, vol. 4, pp. 3572–3575 (1995)
5. Calinescu, R., Ghezzi, C., Johnson, K., Pezzé, M., Rafiq, Y., Tamburrelli, G.: Formal verification with confidence intervals to establish quality of service properties of software systems. IEEE Trans. Reliab. **65**(1), 107–125 (2015)
6. Calinescu, R., Imrie, C., Mangal, R., Păsăreanu, C., Santana, M.A., Vázquez, G.: Discrete-event controller synthesis for autonomous systems with deep-learning perception components. arXiv preprint arXiv:2202.03360 (2022)
7. Calinescu, R., Johnson, K., Paterson, C.: FACT: a probabilistic model checker for formal verification with confidence intervals. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 540–546. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_32
8. Ciesinski, F., Größer, M.: On probabilistic computation tree logic. In: Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P., Siegle, M. (eds.) Validation of Stochastic Systems. LNCS, vol. 2925, pp. 147–188. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24611-4_5

9. Clevert, D.A., Unterthiner, T., Hochreiter, S.: Fast and accurate deep network learning by exponential linear units (ELUs). arXiv preprint arXiv:1511.07289 (2015)
10. Cohen, J., Rosenfeld, E., Kolter, Z.: Certified adversarial robustness via randomized smoothing. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 1310–1320. PMLR, 09–15 June 2019
11. Dawson, C., Gao, S., Fan, C.: Safe control with learned certificates: a survey of neural Lyapunov, barrier, and contraction methods. arXiv preprint arXiv:2202.11762 (2022)
12. Dawson, C., Lowenkamp, B., Goff, D., Fan, C.: Learning safe, generalizable perception-based hybrid control with certificates. IEEE Rob. Autom. Lett. **7**(2), 1904–1911 (2022)
13. Fang, X., Calinescu, R., Gerasimou, S., Alhwikem, F.: Software performability analysis using fast parametric model checking. arXiv preprint arXiv:2208.12723 (2022)
14. Frew, E., et al.: Vision-based road-following using a small autonomous aircraft. In: 2004 IEEE Aerospace Conference Proceedings (IEEE Cat. No.04TH8720), vol. 5, pp. 3006–3015 (2004)
15. Fromherz, A., Leino, K., Fredrikson, M., Parno, B., Pasareanu, C.: Fast geometric projections for local robustness certification. In: International Conference on Learning Representations (2021)
16. Ghosh, S., Pant, Y.V., Ravanbakhsh, H., Seshia, S.A.: Counterexample-guided synthesis of perception models and control. In: 2021 American Control Conference (ACC), pp. 3447–3454. IEEE (2021)
17. Gopinath, D., Converse, H., Pasareanu, C., Taly, A.: Property inference for deep neural networks. In: International Conference on Automated Software Engineering (ASE), pp. 797–809. IEEE (2019)
18. Gopinath, D., Katz, G., Păsăreanu, C.S., Barrett, C.: DeepSafe: a data-driven approach for assessing robustness of neural networks. In: Lahiri, S.K., Wang, C. (eds.) ATVA 2018. LNCS, vol. 11138, pp. 3–19. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01090-4_1
19. Gowal, S., et al.: On the effectiveness of interval bound propagation for training verifiably robust models. arXiv preprint arXiv:1810.12715 (2018)
20. Grigorescu, S.M., Trasnea, B., Cocias, T.T., Macesanu, G.: A survey of deep learning techniques for autonomous driving. CoRR abs/1910.07738 (2019)
21. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. CoRR abs/1706.04599 (2017)
22. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The probabilistic model checker Storm. Int. J. Softw. Tools Technol. Transfer **24**(4), 589–610 (2022)
23. Hoffmann, G.M., Tomlin, C.J., Montemerlo, M., Thrun, S.: Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In: American Control Conference, ACC 2007, New York, NY, USA, 9–13 July 2007, pp. 2296–2301. IEEE (2007)
24. Hsieh, C., Li, Y., Sun, D., Joshi, K., Misailovic, S., Mitra, S.: Verifying controllers with vision-based perception using safe approximate abstractions. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **41**(11), 4205–4216 (2022)
25. Huang, X., et al.: A survey of safety and trustworthiness of deep neural networks: verification, testing, adversarial attack and defence, and interpretability. Comput. Sci. Rev. **37**, 100270 (2020)

26. Huang, X., Kwiatkowska, M., Wang, S., Wu, M.: Safety verification of deep neural networks. CoRR abs/1610.06940 (2016)
27. Ivanov, R., Carpenter, T., Weimer, J., Alur, R., Pappas, G., Lee, I.: Verisig 2.0: verification of neural network controllers using Taylor model preconditioning. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12759, pp. 249–262. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81685-8_11
28. Ivanov, R., Carpenter, T.J., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verifying the safety of autonomous systems with neural network controllers. ACM Trans. Embedded Comput. Syst. (TECS) **20**(1), 1–26 (2020)
29. Ivanov, R., Jothimurugan, K., Hsu, S., Vaidya, S., Alur, R., Bastani, O.: Compositional learning and verification of neural network controllers. ACM Trans. Embedded Comput. Syst. (TECS) **20**(5s), 1–26 (2021)
30. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control, pp. 169–178 (2019)
31. Kadron, I.B., Gopinath, D., Pasareanu, C.S., Yu, H.: Case study: analysis of autonomous center line tracking neural networks. In: Bloem, R., Dimitrova, R., Fan, C., Sharygina, N. (eds.) Software Verification - 13th International Conference, VSTTE 2021, New Haven, CT, USA, 18–19 October 2021, and 14th International Workshop, NSV 2021, Los Angeles, CA, USA, 18–19 July 2021, Revised Selected Papers. LNCS, pp. 104–121 (2021). https://doi.org/10.1007/978-3-030-95561-8_7
32. Katz, G., et al.: The marabou framework for verification and analysis of deep neural networks. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 443–452. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_26
33. Katz, S.M., Corso, A.L., Strong, C.A., Kochenderfer, M.J.: Verification of image-based neural network controllers using generative models. J. Aerosp. Inf. Syst. **19**(9), 574–584 (2022)
34. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47
35. Leino, K., Wang, Z., Fredrikson, M.: Globally-robust neural networks. In: International Conference on Machine Learning (ICML) (2021)
36. Habeeb, P., Deka, N., D'Souza, D., Lodaya, K., Prabhakar, P.: Verification of camera-based autonomous systems. IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., 1 (2023). https://doi.org/10.1109/TCAD.2023.3240131
37. Pasareanu, C.S., et al.: Closed-loop analysis of vision-based autonomous systems: a case study. CoRR abs/2302.04634 (2023). https://doi.org/10.48550/arXiv.2302.04634
38. Privault, N.: Discrete-Time Markov Chains, pp. 77–94. Springer, Heidelberg (2013)
39. Raghunathan, A., Steinhardt, J., Liang, P.: Certified defenses against adversarial examples. In: International Conference on Learning Representations (2018)
40. Santa Cruz, U., Shoukry, Y.: NNLander-VeriF: a neural network formal verification framework for vision-based autonomous aircraft landing. In: Deshmukh, J.V., Havelund, K., Perez, I. (eds.) NASA Formal Methods Symposium, pp. 213–230. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-06773-0_11
41. Singh, G., Gehr, T., Püschel, M., Vechev, M.: An abstract domain for certifying neural networks. Proc. ACM Program. Lang. **3**(POPL), 1–30 (2019)
42. Tabernik, D., Skocaj, D.: Deep learning for large-scale traffic-sign detection and recognition. CoRR abs/1904.00649 (2019)