# Quantifying Assurance in Learning-Enabled Systems

Erfan Asaadi, Ewen Denney, and Ganesh Pai(✉)

KBR, Inc., NASA Research Park, Moffett Field, CA 94035, USA
{easaadi,edenney,gpai}@sgt-inc.com

**Abstract.** Dependability assurance of systems embedding machine learning (ML) components—so called *learning-enabled systems* (LESs)—is a key step for their use in safety-critical applications. In emerging standardization and guidance efforts, there is a growing consensus in the value of using assurance cases for that purpose. This paper develops a quantitative notion of assurance that an LES is dependable, as a core component of its assurance case, also extending our prior work that applied to ML *components*. Specifically, we characterize LES assurance in the form of *assurance measures*: a probabilistic quantification of confidence that an LES possesses system-level properties associated with functional capabilities and dependability attributes. We illustrate the utility of assurance measures by application to a real world autonomous aviation system, also describing their role both in *i)* guiding high-level, runtime risk mitigation decisions and *ii)* as a core component of the associated *dynamic assurance case*.

**Keywords:** Assurance · Autonomy · Confidence · Learning-enabled systems · Machine learning · Quantification

## 1 Introduction

The pursuit of developing systems with increasingly autonomous capabilities is amongst the main reasons for the emergence of *learning-enabled systems* (LESs), i.e., systems embedding machine learning (ML) based software components. There is a growing consensus in autonomy standardization efforts [1] on the value of using *assurance cases* (ACs) as the mechanism by which to convince various stakeholders that an LES can be relied upon. ACs have been successfully used for safety assurance of novel aviation applications where—like LESs—regulations

This work was supported by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under contract FA8750-18-C-0094 of the Assured Autonomy Program. The opinions, findings, recommendations or conclusions expressed are those of the authors and should not be interpreted as representing the official views or policies of DARPA, AFRL, the Department of Defense, or the United States Government.


A. Casimiro et al. (Eds.): SAFECOMP 2020, LNCS 12234, pp. 270–286, 2020.
https://doi.org/10.1007/978-3-030-54549-9_18

and standards continue to be under development [2]. However, LESs pose particular assurance challenges [3] and existing AC technologies may not be sufficient, requiring a framework where the system and its AC evolve in tandem [4]. Here too, there are specific additional challenges: first, structured arguments[1] in many ACs are effectively *static*, i.e., they are usually developed prior to system deployment under assumptions about the environment and intended system behavior. Evolution of the system or its ML components (e.g., via online learning, or by adaptation in operation) can render invalid a previously accepted AC. In principle, although it is possible to dynamically evolve structured arguments [4], since their role is primarily to convince human stakeholders, it makes more sense for such updates to happen between missions at well-defined points.

Second, an operational evaluation of the extent of assurance in an LES (or its ML components, where appropriate) is a valuable system-level indicator of continued fitness for purpose. That, in turn, can facilitate potential intervention and counter-measures when assurance drops below an acceptable level during a mission. Indeed, *online assurance updates* that are aimed at machine consumption must necessarily be in a computable form, e.g., using a formal language, such as a logic, or as a quantification. So far as we are aware, prevailing notions of ACs do not yet admit such evaluation. Prior efforts at AC confidence assessment [5,6] have focused on the argument structure rather than the system itself, and face challenges in repeatable, objective validation due to their reliance on subjective data. They have also not been applied to LESs. Thus, there is a general need to capture a computable form of assurance to bolster an otherwise qualitative AC. Note that although a qualitative AC may well refer to quantitative evidence items, here we are identifying the necessity to have quantified assurance as a core facet of LES ACs.

This paper focuses on the problem of assurance quantification, deferring its use in dynamic updates to future work. The main contribution is an approach to characterize assurance in an LES through uncertainty quantification (UQ) of system-level dependability attributes, demonstrated by application to an aviation domain LES.

## 2 Methodology

Previously [7], we have described how assurance of ML *components* in an LES can be characterized through UQ of component-level properties associated with the corresponding (component-level) dependability attributes. Here, we extend our methodology to the system-level, relying on the following concepts: *assurance* is the provision of (justified) confidence that an *item*—i.e., a (learning-enabled) component, system, or service—possesses the relevant assurance properties. An *assurance property* is a logical, possibly probabilistic characteristic associated

[1] The systematic reasoning that captures the rationale why specific conclusions, e.g., of system safety, can be drawn from the evidence supplied.

with *dependability attributes* [8] and functional capabilities. One or more assurance properties applied to a particular item give an *assurance claim*[2]. An *assurance measure* characterizes the extent of confidence that an assurance property holds for an item through a probabilistic quantification of uncertainty. It can be seen as implementing a UQ model on which to query the confidence in an assurance property.[3]

In general, we can define multiple assurance properties (and assurance measures), based on the LES functionality and dependability attributes for which assurance is sought. For example, the proposition "*the aircraft location does not exceed a specified lateral offset from the runway centerline during taxiing*" is a system-level assurance claim associated with the attribute of *reliability*. Similarly, the assurance property "*the aircraft does not veer off the sides of the runway during taxiing*" is associated with the attribute of *system safety*. Such assurance properties directly map to the claims made in the structured arguments of an LES assurance case. Thus, we can leverage the methodology for creating structured arguments [9] to also specify assurance properties.

For quantification, we mainly consider assurance measures for those system-level properties that can be reasonably and feasibly quantified. For example, assurance measures for the preceding example quantify the uncertainty that the aircraft location does not exceed, respectively, the specified lateral offset from the runway centerline (reliability), and half the width of the runway pavement (safety), over the duration of taxiing.

LESs used in safety-critical applications, especially aviation, are effectively stochastic dynamical systems. The insights from this observation are that we can: *i)* capture LES behavior through model-based representations of the underlying stochastic process; *ii)* view system-level assurance properties as specific realizations of particular random variables (RVs) of that process; and *iii)* express confidence in the assurance properties—i.e., the assurance measures—by propagating uncertainty through the model to determine the distributions over the corresponding RVs.

One challenge is selecting an appropriate model and representation of the stochastic process to be used to model LESs. Although there is not a generic answer for this, such a model could be built, for example, by eliciting the expected system behavior from domain experts, by transforming a formal system description, using model fitting and statistical optimization techniques applied to (pre-deployment) system simulation and execution traces, or through a combination of the three. For LESs, a formal system description may be often unavailable. As such, we rely on elicitation and statistical techniques, using Bayesian models where possible, making allowance to admit and use other well-known, related stochastic process models—such as Markov chains—and leveraging data from analytical representations of system dynamics, simulations, and execution. The Bayesian concepts of *credible* intervals and regions—determined on the

[2] Henceforth, we do not distinguish assurance properties from assurance claims.

[3] When the assurance property is itself probabilistic, the corresponding assurance measure is deterministic, i.e., either 0 or 1.

posterior distribution of the RVs for assurance properties—give a formal footing to the intuitive, subjective notion of confidence that usually accompanies claims in assurance arguments, and ACs in general [10].

## 3 Illustrative Example – Runway Centerline Tracking

**System Description.** To show our methodology is feasible, we now apply it to quantify assurance in an aviation domain LES supplied by our industrial collaborators: a unmanned aircraft system (UAS) embedding an ML component, trained offline using supervised learning, to support an autonomous taxiing capability. The broader goal is to enable safe aircraft movement on a runway without human pilot input. Figure 1 shows a simplified *pipeline architecture* used to realize this capability. A deep convolutional neural network (CNN) implements a perception function that ingests video images from a wing-mounted camera pointed to the nose of the aircraft. The input layer is (360 × 200) pixels ×3 channels wide; the network size and complexity is of the order of 100 layers with greater than two million tunable parameters. Effectively, this ML component performs regression under supervised learning producing estimates of *cross track error* (CTE)[4] and *heading error* (HE)[5] as output. These estimates are input to a classical proportional-integral-derivative (PID) controller that generates the appropriate steering and actuation signals.

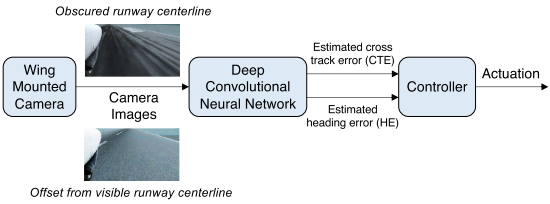


**Fig. 1.** Pipeline architecture to implement an autonomous taxiing capability in a UAS.

### 3.1 Assurance Properties

The main objective during taxiing (autonomously, or under pilot control) is to safely follow the runway (or taxiway) centerline. Safety during taxiing entails avoiding *lateral runway overrun*, i.e., not veering off the sides of the runway pavement. Although avoiding obstacles on the runway is also a safety concern,

[4] The horizontal distance between the aircraft nose wheel and the runway centerline.
[5] *Heading* refers to the compass direction in which an object is pointed; heading error (HE) here, is thus the angular distance between the aircraft heading and the runway heading.

it is a separate assurance property that we do not consider in this paper. Thus, safety can be achieved here, in part, by meeting a performance objective of maintaining an acceptable lateral offset (ideally zero) on either side of the runway centerline during a taxi *mission* from starting taxi to stopping (or taking off).[6] In other words, the closer the aircraft is to the runway centerline during taxiing, the less likely it is to veer off the sides of the runway.

This performance objective relates to the attribute of reliability, where *taxi failure* is considered to be the violation of the specified lateral offset. Here, we focus on the corresponding assurance property, `AssuredTaxi` : $|\text{CTE}_a| <$ `offset`, where `offset` $= 2\,\text{m}$ is the maximum acceptable lateral offset on either side of the runway centerline for this application and aircraft type. $\text{CTE}_a$, which is the true (or actual) CTE for the UAS, is a signed, real valued scalar; the absolute value gives the magnitude of the offset, and the sign indicates where the UAS is located relative to the centerline, i.e., to its left or its right.

### 3.2 Assurance Quantification

**Model Choice.** The assurance measure corresponding to `AssuredTaxi`, establishes $\Pr(|\text{CTE}_a| < 2\,\text{m})$, which characterizes the uncertainty (or conversely, confidence) in the true (or actual) CTE ($\text{CTE}_a$) relative to the specified offset. $\text{CTE}_a$ evolves in time as the PID controller responds to *estimates* of CTE and HE, themselves the responses of the deep CNN component, to runway images captured by the wing mounted camera (see Fig. 1). $\text{CTE}_a$ is thus uncertain and depends on other variables, of which those that can be observed are the estimated CTE ($\text{CTE}_e$), estimated HE ($\text{HE}_e$), and a sequence of images. We can also model the controller behavior in terms of a time series evolution of $\text{CTE}_a$ since, during taxiing, the true CTE at a given time is affected by the controller actuation signals at prior times.

An abstracted model of LES behavior is reflected in the joint distribution of the relevant observed and uncertain variables. In fact, a *dynamic Bayesian network* (DBN) [11] is a convenient and compact representation of this joint distribution, as we will see subsequently in this section. It takes into account the temporal evolution of the variables and their (known or assumed) conditional independence relations. Thus, to determine the assurance measure, we effectively seek to quantify the (posterior) distribution over $\text{CTE}_a$, given a sequence of runway images, the estimates of CTE and HE produced by the ML component, and the controller behavior, as a query over the corresponding DBN model.

**Model Variables.** Model variables can be discrete or continuous, and there are tradeoffs between information loss and computational cost involved in the choice. Table 1 lists the discrete variables we have chosen, giving the interval boundaries for their states. The choice of the intervals that constitute the states of the variables has been based, in part, on: *i)* domain knowledge, *ii)* an assessment

[6] Our industry collaborators elicited the exact performance objectives from current and proficient professional pilots.

**Table 1.** DBN model variables.

| Description | Variable | Interval Boundaries for States |
|---|---|---|
| True CTE<br>*Uncertain discrete variable with 9 states* | $CTE_a$ | [-½$w$, -2, -1.43, -0.85, -0.28, 0.28, 0.85, 1.43, 2, ½$w$] |
| Outlier detection outcome<br>*Observed Boolean variable* | D | [0, 1] |
| CNN estimate of CTE<br>*Observed discrete variable with 6 states* | $CTE_e$ | [-½$w$, -2, -1, 0, 1, 2, ½$w$] |
| CNN estimate of HE<br>*Observed discrete variable with 3 states* | $HE_e$ | [-20, -3, 3, 20] |

of the data sampled from the environments used for training and testing the CNN, and *iii)* the need to develop an executable model that was modest in its computational needs.

Here, $w$ is the width of the runway in meters, and negative values represent CTE measured on the left of the runway centerline. The HE is given in degrees, while $D$ is dimensionless. An additional variable ($I$, not shown in Table 1) models the runway image captured from the camera video feed as a vector of values in the range $[0 \ldots 1]$ representing normalized pixel values. The Boolean variable $D$ represents the detection of outliers in camera image data. Such outliers may manifest due to various causes, including camera errors and *covariate shift*, i.e., when the data input to the CNN has a distribution different from that of its training data. Note that the LES shown in Fig. 1 does not indicate whether or not it includes a mechanism to detect outliers or covariate shift. However, we include this variable here, motivated by our earlier work on component-level assurance quantification of the CNN [7], which revealed its susceptibility to outlier images. In fact, $D$ models a runtime monitor for detecting out of distribution (OOD) inputs to the CNN.

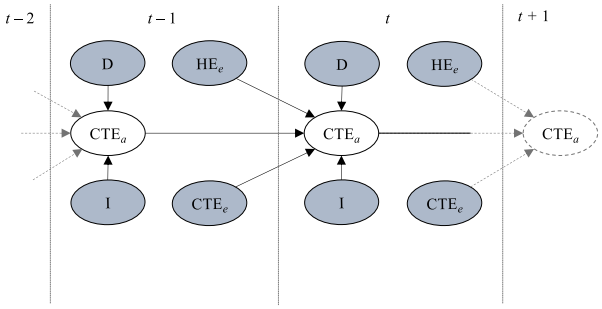


**Fig. 2.** DBN structure for assurance quantification, showing two adjacent slices at times $t-1$, and $t$; shaded nodes represent observed variables, clear nodes are the uncertain, latent variables.

**Model Structure.** Each variable in Table 1 is indexed over time: we will denote a variable $X$ at time $t$ as $X^{(t)}$. The causal ordering of the model variables (Fig. 2) informs the structure of the DBN: the estimated CTE and HE at time $t$ are inputs to the controller which, in turn, impacts the future location of the aircraft at time $t + \varepsilon$. The directed links between the corresponding variables in adjacent time slices capture this dependency. For example, in Fig. 2, these are the directed links $\text{CTE}_e^{(t-1)} \rightarrow \text{CTE}_a^{(t)}$, and $\text{HE}_e^{(t-1)} \rightarrow \text{CTE}_a^{(t)}$ (and likewise for the preceding and subsequent time slices). The directed links $\text{CTE}_a^{(t-1)} \rightarrow \text{CTE}_a^{(t)}$ model the correlation between actual vehicle position over time, also capturing vehicle inertia.

At time $t$, the runway image $I^{(t)}$ influences the belief about the true aircraft location, i.e., the states of $\text{CTE}_a^{(t)}$, with the node $D$ modeling the associated structural uncertainty. This reflects the intuition that upon detecting an outlier image (more generally an OOD input), we are no longer confident that the image seen is an indicator of the actual aircraft location. Figure 2 reflects these dependencies by the directed edges $\text{CTE}_a^{(t)} \leftarrow I^{(t)}$, and $\text{CTE}_a^{(t)} \leftarrow D^{(t)}$, respectively.

Figure 2 shows two adjacent time slices of the DBN structure, although the actual structure is unrolled for $T$ time steps, the duration of taxiing, to compute the assurance measure over the taxi phase. At time $t$, this is, in fact, the sum of the probability mass over the seven states of $\text{CTE}_a^{(t)}$ that lie within the interval $[-2, 2]$ (see Table 1). By unrolling the DBN for an additional $\varepsilon$ time steps and propagating the uncertainty through the model from the time of the last observations, the model can provide an assurance forecast.

**Probability Distributions.** To complete the DBN model specification, we need to specify the conditional probability distributions (CPDs) over the model variables, as encoded by its structure. One way to identify the CPDs is through uncertainty quantification of the physical system model [12]. Practically, the latter may not be available, especially for LESs.

Another alternative—the approach we take here—is to assume a functional form for the CPDs that is then tuned based on execution and simulation data. Specifically, to construct the CPD represented by the transition edge between the time slices, i.e., $\Pr(\text{CTE}_a^{(t)} \mid \text{CTE}_a^{(t-1)}, \text{CTE}_e^{(t-1)}, \text{HE}_e^{(t-1)})$, we chose a multinomial distribution with a uniform prior, tuned using the maximum a posteriori probability (MAP) estimate on simulation data. This choice was advantageous in the sense that the DBN produces a uniform posterior distribution over $\text{CTE}_a$ when the observed variables take on values from a distribution different from that of the data used to build the CPDs. For this example, the simulation data comprised sequences of runway images, estimated CTE and HE as produced by the CNN, and true CTE. Section 4 gives more details on the simulation platform and data gathered.

To determine the emission probability $\Pr(\text{CTE}_a^{(t)} \mid I^{(t)})$, first we used the Gaussian process (GP) model underpinning our prior work on component-level assurance quantification [7]. In brief, the idea is to use a GP to model the error performance of the CNN (i.e., its accuracy) on its input (i.e., runway images).

Then, adding the error distribution to the estimate of CTE gives the distribution over the true CTE. However, for high dimensional data (such as images), this is computationally expensive. Instead, in this paper we used an ensemble of decision trees [13] as a classifier that ascribes a probability distribution over the states of $\text{CTE}_a$, given a runway image, $I$. This approach builds uncorrelated decision trees such that their combined estimate is more accurate than that of any single decision tree. To identify the decision rules, we used supervised learning over the collection of runway images and corresponding true CTE, sampled from the same environments used to train and test the CNN (see Sect. 4). For this example, we built 280 decision trees with terminal node size of at least 10, by randomly sampling 100 data points using the *Gini index* as a performance metric, selecting the model parameters to balance classification accuracy and computational resources.

## 4 Experimental Results

We now present some results of our experiments in quantifying LES assurance in terms of the assurance measure, $\Pr(|\text{CTE}_a| < 2\,\text{m})$, based upon simulations of constant speed taxiing missions.

**Simulation Setup.** We use a commercial-off-the-shelf flight simulator instrumented to reflect the pipeline architecture of Fig. 1. The simulation environment includes various airports and runways with centerlines of varying quality, e.g., portions of the centerline may be obscured at various locations (see Fig. 1). We can create different training and test environments by changing various simulation settings, among which two that we have selected are: *i)* weather induced visibility (*clear* and *overcast*), and *ii)* the time of day (*07:30 am* to *2:00 pm*). Two such environments are, for example, "*Clear at 07:30 am*", and "*Overcast at 12:15 pm*". More generally, we can construct environments such as "*Clear Morning*", "*Overcast Afternoon*", and so on. The former refers to the collection of data sampled from the environment having clear weather, and the time of day incremented in steps of 15 and 30 min from *07:30 am* until *noon*. A similar interpretation applies to other such environments.

From these environments, we gathered images via automated screen capture (simulating the camera output) whilst taxiing the aircraft on the airport runway, using different software controllers, as well as different CNNs for perception: i.e., the same CNN architecture described in Sect. 3, but trained by our industrial collaborators with data drawn from the various environments identified earlier. In tandem, for each image, we collected true CTE (from internal simulation variables), along with estimates of CTE and HE. We used several such data sets, one for each of the different environments identified above, from which data samples were drawn to build the CPDs of the DBN model. Here, note that these data samples were *not* identical to those used to train and test the CNN, even though the samples were drawn from the collection of environments common to both the LES and the DBN.

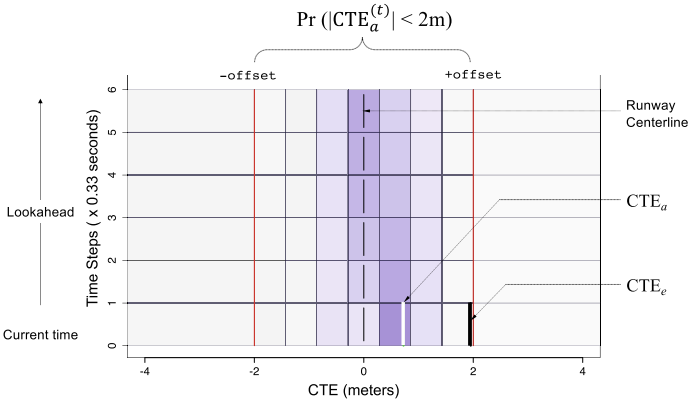


**Fig. 3.** Visualization of predicted uncertainty in true cross track error, $\text{CTE}_a^{(t)}$, to quantify assurance in runway centerline tracking as the assurance measure, Pr(`AssuredTaxi`).

**Uncertainty Quantification.** Figure 3 shows the results of assurance quantification for one test scenario, visualized as a probability surface overlaid on a stretch of the runway, itself shown as a grid. The horizontal axis—discretized using the interval boundaries for the states of $\text{CTE}_a$ (see Table 1)—gives the true aircraft location, which is uncertain during taxiing. Thus, moving from left to right (or vice versa) constitutes lateral aircraft movement. The vertical axis (discretized into 6 steps, each of duration 0.33 s) represents the number of time slices for which the DBN model is unrolled. We selected this based on the time taken for the UAS to laterally depart the runway after violating the 2 m bound, given: runway dimensions, maximum allowed taxiing speed, and other constraints on the UAS dynamics, e.g., non-accelerating taxiing.

At $t = 0$, the horizontal axis gives the aircraft location at the current time. The time steps $t = 1, \ldots, 6$ are *lookahead times* for which the horizontal axis gives the *predicted* location of the aircraft relative to the centerline, given the CNN estimates of CTE and HE at $t = 0$. Thus, moving from the bottom to the top of Fig. 3 represents forward taxiing, i.e., the temporal evolution of aircraft position over the runway. Each cell of the grid formed by discretizing the two axes is, therefore, a state of $\text{CTE}_a$ at a given time, shaded such that darker shades

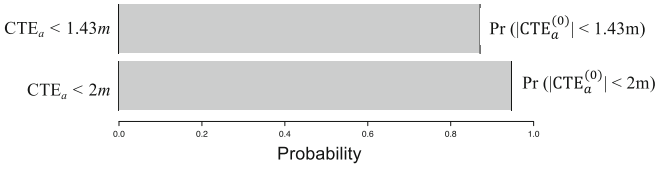


**Fig. 4.** Pr(`AssuredTaxi`) for `offset` = 2 m and `offset` = 1.43 m.

indicate lower uncertainty (or higher confidence) and lighter shades indicate higher uncertainty (or lower confidence). Thus, the row at $t = 0$ shows the DBN estimate of uncertainty over $\text{CTE}_a$ at the current time. Similarly, each row for $t = 1, \ldots, 5$ shows the *predicted* uncertainty over $\text{CTE}_a$ for those lookahead times, given that the last known values for the observed variables are at $t = 0$. The solid white line in Fig. 3 at $t = 0$ is *ground truth*, i.e., the true CTE at the current time based on internal simulation variables. Although this may not be otherwise available during taxiing, we show it here primarily for model validation, i.e., to show that the interval (state of $\text{CTE}_a$) estimated by the DBN to be the least uncertain is also the one that includes the ground truth. The solid black line is CTE as estimated by the CNN (i.e., $\text{CTE}_e$) at the current time.

Recall that assured taxiing involves maintaining $\text{CTE}_a$ between a 2 m lateral offset on either side of the centerline. To quantify assurance in this property, we sum up the probability mass in each cell between the two offsets. Figure 4 shows the assurance measure, $\Pr(|\text{CTE}_a^{(t=0)}| < \texttt{offset})$ computed for two different offset values: 2 m and 1.43 m.[7] The interval $[-2, 2]$ is a Bayesian *credible interval* within which the true CTE lies with probability $\approx 95\%$, based on Fig. 4. In other words, *the DBN model is* $\approx$95% *confident that the aircraft is truly located within* 2 m *of the runway centerline*. In general, the expected (and desired) DBN behavior is to be more uncertain over longer term assurance forecasts, when there are no additional observations with which to update the posterior distributions on the assurance measures.

**Sufficient Assurance.** We must select a threshold on the assurance measure to establish what sufficient assurance constitutes, based on which we can assert whether or not the assurance claim holds. The criterion we have selected here is: when the DBN is $\geq$30% confident that the true UAS location exceeds the allowed lateral offset, the assurance claim does not hold, i.e., $\Pr(|\text{CTE}_a^{(t)}| \geq 2\,\text{m}) \geq 0.3 \Rightarrow \neg(\texttt{AssuredTaxi})$. We determined this threshold under conservative assumptions about vehicle behavior, leveraging the engineering judgment of our industry collaborators, to balance the tradeoff between safety (avoiding runway overrun) and mission effectiveness (not stopping too often).

## 5 Discussion

We now evaluate how the DBN performs relative to the LES, in the context of ground truth. The intent is to show that it is a reasonable (i.e., valid) *reference model* of the system suitable for runtime use (i.e., simple and abstract), based on which to make certain decisions, e.g., whether or not to stop taxiing. Moreover, we must also show that the software implementation of the DBN can be relied upon. In this paper, we primarily address the former, leaving the latter for future work.

[7] The introduction of a second offset was motivated by our industry collaborators to integrate the assurance measure on the LES platform.

**Validity.** We compare how well the DBN and the LES can discriminate between *true positive* and *true negative* situations when their respective outputs are transformed into a classification on a plurality of image data drawn from multiple simulated taxiing scenarios for different test environments unseen by both the DBN and the LES.

A true positive (negative) situation for the DBN is one where it indicates that the assurance property is satisfied (not satisfied) based on the criterion for sufficient assurance (see Sect. 4), and ground truth data also indicates that it is truly the case that the UAS location is within (exceeds) the allowed lateral offset from the runway centerline. Likewise for the LES, a true negative (positive) situation is one where the CNN estimate of CTE indicates (does not indicate) an offset violation i.e., $\text{CTE}_e \geq 2\,\text{m}$ (equivalently, $\text{CTE}_e < 2\,\text{m}$), and so does ground truth data.

**Table 2.** DBN Performance evaluation for runway centerline tracking.

| Perception Component | Test Environment | LES Performance | | DBN Model Performance | |
|---|---|---|---|---|---|
| | | Sensitivity | Specificity | Sensitivity | Specificity |
| CNN trained on *Clear Afternoon* and *Overcast Afternoon* environments | *Clear at 07:30am* | 0.85 | 0.95 | 1 | 1 |
| | *Clear at 10:15am* | 0.83 | 0.87 | 1 | 1 |
| | *Overcast at 07:30am* | 0.87 | 0.85 | 1 | 1 |
| CNN trained on *Clear Morning* and *Clear Afternoon* environments | *Overcast at 12:15pm* | 1 | 0.01 | 0.8 | 0.99 |
| | *Clear at 11:45am* | 1 | 0.2 | 0.98 | 0.8 |
| | *Clear at 07:30am* | 1 | NA | 1 | NA |

Table 2 shows our evaluation results in terms of *sensitivity* (true positive rate) and the *specificity* (true negative rate) of both the DBN model and the LES, varying the embedded CNN used for perception. The variability arises from using CNNs trained under two different training environments. We also used these training environments to build the DBN for both LES variants using ≈37000 image samples. These samples were not the same as those that were used to train the CNN variants: indeed, we did not have access to the actual training data for the different CNNs. Also, the test environments listed in the table (and, therefore, the resulting test data), are unseen during the development of both LES variants, and the DBN models of the same.

Based on Table 2, in the context of the sensitivity and specificity metrics shown, as well as the criterion for sufficient assurance, we are cautiously optimistic in claiming that the DBN models the LES reasonably well. For the test environments "*Clear at 11:45 am*", and "*Overcast at 12:15 pm*", the DBN has a lower sensitivity than the LES, however its specificity is substantially better. This suggests that the LES may be biased in its estimates of CTE for those operating conditions.

**Suitability.** The DBN model structure—in particular, the conditional independence relations encoded by the structure—is informed by (our knowledge of) the causal impacts of the identified variables and the system dynamics, and the resulting assumptions. We note that it is always possible to relax these assumptions and learn the DBN structure as well as its parameters. However, in most cases, especially when there is limited data available, structure learning can be an unidentifiable problem, or can produce a non-unique solution. In our case, the conditional independence assumptions used have turned out to be neither too strong to affect model performance nor too conservative to impose a problem in identifying the CPDs given limited data.

Our assessment in Table 2 does *not* compare the DBN and the CNN that estimates CTE. Indeed, the latter is a learned, static regression function for a *component*, that associates a vector of real values with a real-valued scalar, whereas here we are assessing a stochastic process model of a (learning-enabled) *system* (i.e., the DBN) against the system itself. When we use the DBN for runtime assurance, we implement it as a software component integrated into the LES. This can be viewed as an item to which we can apply our own assurance methodology, i.e., as in Sect. 2, and [7]. Thus, although we have not formulated assurance properties for the DBN, sensitivity and specificity are probabilistic performance metrics (albeit in a frequentist sense) that we can view as assurance measures in their own right, that we have now applied to our model.

The validation above is admittedly not exhaustive although the following observations are worth noting: the DBN is a relatively simple and abstract model of the time-series evolution of the *system*, whose estimates can be updated through Bayesian inference given observed data. Thus, it is amenable to applying other verification techniques including inspection, and formal verification.

Moreover, the DBN *does not* produce point estimates of CTE; rather, in quantifying confidence in a system-level assurance property, a by-product is the uncertainty in true CTE given as a probability distribution over the range of admissible values of $\text{CTE}_a$. Thus, in unseen situations where the CNN can produce an inaccurate estimate of CTE (see Fig. 3), the DBN gives a distribution over possible values of true CTE. As such, it is more conservative in potentially unsafe scenarios. Based on this assessment, we submit that the DBN is a reasonable and suitable runtime reference model of the LES for the autonomous taxiing application, when used for centerline tracking.

**Utility.** A key advantage of an abstract assurance quantification model is a small implementation footprint for runtime integration into the LES. As indicated in Sect. 1, one of the primary motivations for quantified assurance measures is to provide feedback signals (in a computable form) to the LES, that can be acted on, e.g., by a *Contingency Management System* (CMS), in operation. In this work, the assurance measure values were translated into commands to either *stop*, *slow down*, or *continue* based on *i)* the chosen decision thresholds (Sect. 4), and *ii)* a simple model of the system-level effect (i.e., likelihood of lateral runway

overrun) given the assurance measure and current system state.[8] In general, deciding between a series of options in the presence of conflicting and uncertain outcomes is a special case of *decision making under uncertainty* [14]. We plan to investigate such techniques as future work to develop a principled approach to contingency management using assurance measures.

The aim of *run-time assurance*, also known as *run-time verification*, is to provide updates as to whether a system satisfies specified properties as it executes [15]. This is done using a run-time *monitor*, which evaluates the property using values extracted from the state of the system and its environment. In a sense, therefore, the notion of assurance measure we have described here is a kind of monitor. However, it is worth making several distinctions. A monitor relates directly to properties of the system, whereas an assurance measure characterizes *confidence* in our knowledge of such properties. Second, an assurance measure seeks to aggregate a range of sources of information, including monitors. Thus it can be seen as a form of *data fusion*. Third, monitors typically provide values that relate to the current state of the system, whereas the assurance measures we have defined are predictive, intended to give a probabilistic quantification on dependability attributes.

In general, our approach to assurance quantification admits other models including runtime monitors: recall that the node $D^{(t)}$ in Fig. 2 is a runtime monitor detecting data distribution shift in the input image at time $t$. Indeed, our framework is not intended to replace runtime verification, and the assurance measures generated show the assurance contribution of the runtime monitors, additionally providing an assurance/uncertainty forecast. We are not aware of existing runtime verification techniques that do this.

## 6 Related Work

The work in this paper is closely related to our earlier research on assurance case confidence quantification [5]. There, although confidence estimation in an assurance claim also uses Bayesian techniques, it relies primarily on the argument structure to build the model. Similarly, based on the structure of an argument, the use of an evidential theory basis has been explored for confidence quantification in assurance claims [6]. However, neither work has been applied to LES assurance quantification. Moreover, in this paper the focus is on those properties where quantification is possible, relying upon models of the system that can be assessed against objective, measured data.

This paper is a natural extension of our prior work on quantifying assurance in ML components [7]: the assurance property we consider there is $\text{CTE}_e$ *accuracy*. Assurance quantification then entails using Gaussian processs (GPs) to determine the uncertainty in the error of $\text{CTE}_e$, which is inversely proportional to accuracy. However, the data used are not (and need not be) time dependent

[8] Although the content of integrating assurance measures with a CMS is very closely related to the work here, it is not in scope for this paper, and will be the topic of a forthcoming article.

and the model used applies regardless of whether or not the aircraft position has violated `AssuredTaxi`. Indeed, despite a high assurance CNN that accurately estimates CTE, it is nevertheless possible to violate `AssuredTaxi`. However, in this paper we model the LES as a stochastic process, including any runtime mitigations, e.g., a monitor for detection OOD images. As such, the models used for UQ are a generalization of that in [7] to time-series behavior.

As previously indicated (Sect. 1), one of the motivations is to support dynamic assurance cases (ACs). Our prior work [4] first explored this concept, which has subsequently been tailored for so-called *self-adaptive software* [16]. Again, neither work has considered LESs, although self-adaptation is one of the properties that LESs can exhibit. In [4], confidence quantification has been situated as a core principle of dynamic assurance which has also motivated this paper to an appreciable degree. However, that work relies on the quantification methodology in [5]. In [16], assurance quantification employs probabilistic model checking, which can be leveraged for LESs if they can be represented using state-space models, e.g., as in [17] which uses hybrid model checking instead. Neither technique is incompatible with the stochastic processes-based modeling approach that we have adopted. As such, they may be a candidate means to check properties of the stochastic models that we build as a means of (meta-)assurance.

*Dynamic safety management* as an assurance concept has also been proposed as a run-time assurance method [18], but it is largely speculative about applicability for LESs. The idea of *requirements-aware* runtime models [19] is very closely related to our notion of building a reference model. Quantified and probabilistic guarantees in reinforcement learning have been explored in developing assured ML components in [20]. That work is also closely related to what we have presented here, though its focus is mainly on assurance of correctness properties that have a safety impact. Additionally, the assurance approach there is *intrusive* in the sense that the ML component being built is modified. In our case, assurance quantification does not modify the ML components. *Benchmarking* of uncertainty estimation techniques [21] has also been investigated, although mainly in the context of image classification. It is unclear if the reported results translate to assurance quantification as applied in this paper. However, the benchmarking principles and metrics used could be candidates for evaluating various system models built using our approach.

Kalman filters have long been used to address uncertainty during state estimation, and have some similarities to our approach. A Kalman filters is a special case of a DBN where amongst the main assumptions are that sensor errors are distributed as zero mean Gaussians, and that the uncertainty does not vary between sensing outputs. In contrast, our model uses discrete distributions, admitting varying sensor uncertainty for each image input, in a more general graphical model that has a different structure, whilst including detections of OOD inputs.

## 7 Conclusion and Future Work

We have described our approach to quantifiable assurance using assurance measures, run-time computations of uncertainty (conversely, confidence) in specified assurance properties, and their application to learning-enabled systems (LESs). Assurance measures complement design-time assurance activities, each of which forms part of an overall dynamic assurance case (DAC). In collaboration with system integrators from industry, we have applied our framework to an aviation platform that employed supervised learning using a deep CNN. Collaboration was crucial to develop the contingency management capability, which relied on engineering judgment to tradeoff safety risk reduction and achieving performance objectives. Feedback from the end-users (i.e., our industry collaborators) was also essential in refining the final visualizations of the assurance measure that we ultimately deployed in the system (based on Fig. 4). Those are intended to provide insight into the system assurance state for safety observer crew.

We have shown that our methodology can feasibly quantify assurance in system-level properties of an aviation domain LES, though we have used classical UQ techniques. Our work in quantifying assurance in LESs is ongoing, and we will be developing assurance measures for other autonomous platforms in the context of more complex mission objectives that require additional ML components and learning schemes.

The work in this paper is one strand of our overall approach to assurance through DACs. The diverse components of an assurance case, including structured arguments, safety architecture [22], as well as the assurance measures described here, each represent one facet of an integrated DAC. There are close connections between the probabilistic models underlying assurance measures and the safety architecture, as well as between assurance properties and claims in an assurance arguments. Our future work will place these connections on a rigorous basis. In part, this can be achieved through use of a high-level *domain-specific language* (DSL) that will let us *i)* abstract from the details of the individual probabilistic models, and *ii)* conversely, allow compilation into a range of different models, whilst making more explicit the connections to domain concepts used elsewhere in the assurance case.

A related avenue of future work is providing comprehensive assurance for our approach itself, and in turn, the assurance measures produced. From a verification standpoint, we can consider *correctness* properties entailing *i)* consistency between the quantification model and the other DAC components, e.g., the risk scenarios captured by a safety architecture, and *ii)* correctness of the low-level implementation against the higher level specification embodied by the quantification model.

Additionally, assurance measure validity is related, in part, to the limits of the statistical techniques used to infer the underpinning stochastic models, and the data used to build them.

Indeed, one of the challenges we faced in this work was obtaining sufficient useful data. Moreover, the quality of the data gathered also plays a key role in corroborating that the assurance quantification models sufficiently represent

the system behavior across its intended operational profile. We believe that a more principled approach to specifying a variety of training data should be possible (e.g., to include various types of perturbed and adversarial inputs), and that such specifications could be derived from the DSL used to specify the assurance measures themselves. The dynamic nature of assurance cases (ACs) will also bear further investigation, to see how real-time updates provided by assurance measures during a mission can inform updates between missions, to the qualitative arguments of ACs.

## References


1. Underwriter Laboratories Inc.: Standard for Safety for the Evaluation of Autonomous Products UL 4600, April 2020
2. Clothier, R., Denney, E., Pai, G.: Making a risk informed safety case for small unmanned aircraft system operations. In: 17th AIAA Aviation Technology, Integration, and Operations Conference (ATIO 2017), AIAA Aviation Forum, June 2017
3. McDermid, J., Jia, Y., Habli, I.: Towards a framework for safety assurance of autonomous systems. In: Espinoza, H., et al. (eds.) 2019 AAAI Workshop on Artificial Intelligence Safety (SafeAI 2019), CEUR Workshop Proceedings, January 2019
4. Denney, E., Habli, I., Pai, G.: Dynamic safety cases for through-life safety assurance. In: IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE 2015), vol. 2, pp. 587–590, May 2015
5. Denney, E., Pai, G., Habli, I.: Towards measurement of confidence in safety cases. In: 5th International Symposium on Empirical Software Engineering and Measurement (ESEM 2011), pp. 380–383, September 2011
6. Wang, R., Guiochet, J., Motet, G., Schön, W.: Safety case confidence propagation based on Dempster-Shafer theory. Int. J. Approximate Reasoning **107**, 46–64 (2019)
7. Asaadi, E., Denney, E., Pai, G.: Towards quantification of assurance for learning-enabled components. In: 15th European Dependable Computing Conference (EDCC 2019), pp. 55–62. IEEE, September 2019
8. Avižienis, A., Laprie, J.C., Randell, B., Landwehr, C.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. Dependable Secure Comput. **1**(1), 11–33 (2004)
9. Denney, E., Pai, G.: Tool support for assurance case development. J. Autom. Softw. Eng. **25**(3), 435–499 (2018)
10. Hawkins, R., Kelly, T., Knight, J., Graydon, P.: A new approach to creating clear safety arguments. In Dale, C., Anderson, T. (eds.) Advances in Systems Safety, pp. 3–23 (2011)
11. Murphy, K.P.: Machine Learning: A Probabilistic Perspective. MIT Press, Cambridge (2012)
12. Najm, H.N.: Uncertainty quantification and polynomial chaos techniques in computational fluid dynamics. Annu. Rev. Fluid Mech. **41**(1), 35–52 (2009)
13. Criminisi, A., Shotton, J., Konukoglu, E.: Decision forests: a unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. Found. Trends Comput. Graphics Vision **7**(2–3), 81–227 (2012)

14. Kochenderfer, M.J.: Decision Making Under Uncertainty: Theory and Application. MIT Press, Boston (2015)
15. Moosbrugger, P., Rozier, K.Y., Schumann, J.: R2U2: monitoring and diagnosis of security threats for unmanned aerial systems, pp. 1–31, April 2017
16. Calinescu, R., Weyns, D., Gerasimou, S., Iftikhar, M.U., Habli, I., Kelly, T.: Engineering trustworthy self-adaptive software with dynamic assurance cases. IEEE Trans. Software Eng. **44**(11), 1039–1069 (2018)
17. Ivanov, R., Weimer, J., Alur, R., Pappas, G.J., Lee, I.: Verisig: verifying safety properties of hybrid systems with neural network controllers. In: 22nd ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2019, pp. 169–178 (2019)
18. Trapp, M., Schneider, D., Weiss, G.: Towards safety-awareness and dynamic safety management. In: 14th European Dependable Computing Conference, EDCC 2018, pp. 107–111, September 2018
19. Bencomo, N., Garcia-Paucar, L.H.: RaM: causally-connected and requirements-aware runtime models using Bayesian learning. In: 22nd IEEE/ACM International Conference on Model Driven Engineering Languages and Systems, MODELS 2019, September 2019
20. Bouton, M., Karlsson, J., Nakhaei, A., Fujimura, K., Kochenderfer, M.J., Tumova, J.: Reinforcement learning with probabilistic guarantees for autonomous driving. Computing Research Repository (CoRR) arXiv:1904.07189v2 [cs.RO], May 2019
21. Henne, M., Schwaiger, A., Roscher, K., Weiss, G.: Benchmarking uncertainty estimation methods for deep learning with safety-related metrics. In: Espinoza, H., et al. (eds.) 2020 AAAI Workshop on Artificial Intelligence Safety (SafeAI 2020), CEUR Workshop Proceedings, vol. 2560, pp. 83–90, February 2020
22. Denney, E., Pai, G., Whiteside, I.: The role of safety architectures in aviation safety cases. Reliab. Eng. Syst. Saf. **191**, 106502 (2019)