Authors' Preprint Copy. 15th European Dependable Computing Conference (EDCC 2019). Sep. 2019.
The definitive version of this article appears in the EDCC 2019 Conference Proceedings on IEEExplore.

Towards Quantification of Assurance for Learning-enabled Components

Erfan Asaadi, Ewen Denney, Ganesh Pai
SGT, Inc., NASA Research Park
Moffett Field, CA 94035, USA
{easaadi, edenney, gpai}@sgt-inc.com

***Abstract*—*Perception*, *localization*, *planning*, and *control*, high-level functions often organized in a so-called *pipeline*, are amongst the core building blocks of modern autonomous (ground, air, and underwater) vehicle architectures. These functions are increasingly being implemented using *learning-enabled components* (LECs), i.e., (software) components leveraging knowledge acquisition and learning processes such as deep learning. Providing quantified component-level assurance as part of a wider (dynamic) assurance case can be useful in supporting both pre-operational approval of LECs (e.g., by regulators), and runtime hazard mitigation, e.g., using assurance-based failover configurations. This paper develops a notion of assurance for LECs based on *i*) identifying the relevant dependability attributes, and *ii*) quantifying those attributes and the associated uncertainty, using probabilistic techniques. We give a practical grounding for our work using an example from the aviation domain: an autonomous taxiing capability for an unmanned aircraft system (UAS), focusing on the application of LECs as sensors in the perception function. We identify the applicable quantitative measures of assurance, and characterize the associated uncertainty using a non-parametric Bayesian approach, namely Gaussian process regression. We additionally discuss the relevance and contribution of LEC assurance to system-level assurance, the generalizability of our approach, and the associated challenges.**

***Index Terms*—Assurance, Autonomy, Confidence, Convolutional neural networks, Deep learning, Learning-enabled components, Machine learning, Quantification**

I. Introduction

Modern autonomous air, ground, and underwater vehicle architectures use the high-level functions of *perception* (converting sensed data into a model of the environment in which the vehicle is situated), *localization* (establishing vehicle position and location within its environment), *planning* (finding an optimal suite of actions to achieve goals such as following a path or trajectory, or navigation in general) and *control* (regulating high-level vehicle behavior and low-level actuation as required), as core building blocks typically organized in a so-called *pipeline* [1], i.e., the functions are separate, broadly organized in the above order in a modular way, with the output of one function serving as the input to the next.

Increasingly, some or all of these functions are being implemented using *learning-enabled components* (LECs), i.e., (software) components that leverage knowledge acquisition and machine learning processes. In particular, deep neural networks (DNNs) have enjoyed a wide proliferation owing to the availability of large datasets and advances in graphics processing units (GPUs) [2].

In safety-critical domains such as aviation, before systems can be deployed into civil airspace, assurance must be provided to various stakeholders (chiefly, the regulator) not only that those systems have been designed and developed to be inherently safe, but also that they can be operated at an acceptable level of safety. That, in turn, often also requires assurance that the constituent components perform adequately, are sufficiently reliable, etc.[1] As such, when LECs are being considered for use in aviation—e.g., to realize transformative application concepts such as *urban air mobility* [3]—assurance of their fitness for purpose will inevitably be required both at a component and a system level, as appropriate. This is the primary motivation for the work in this paper, with the focus being on component-level assurance that LECs possess dependability attributes [4] additional to safety.

For novel aviation applications where regulations either do not yet exist, or are still being developed (e.g., in enabling unmanned aircraft to fly beyond visual range), *safety cases* have been a successful means by which to assure regulators of system safety [5]. An *assurance case* (AC) is a generalization of the safety case concept meant to provide justified confidence that a system is *fit for purpose*, and it addresses the broader system attributes, e.g., of dependability. The applicability of ACs is now being progressively explored for learning-enabled systems (LESs)—i.e., systems containing LECs—most notably for safety assurance of self-driving road vehicles [6].

For the most part, ACs have used *structured arguments*[2] as the mechanism to engender justified confidence. Our prior work has advanced a notion of *dynamic safety case* (DSC) [7]—more generally, *dynamic assurance case* (DAC)—wherein a central tenet was using *confidence quantification* [8] to provide assurance, in addition to structured arguments. The work in this paper, which we cast within this broader DAC framework, has two aspects that distinguish it from our prior work: firstly, we focus on component-level assurance

The Defense Advanced Research Projects Agency (DARPA) has supported this work under contract FA8750-18-C-0094 of the Assured Autonomy Program. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

[1] In fact, components used on civil aircraft are required to meet prescribed minimum performance standards known as *technical standard orders* (TSOs).

[2] A chain of reasoning that conveys the rationale why certain conclusions can be drawn (e.g., of acceptable safety) based upon the evidence supplied.

quantification to be used within a wider DAC. Second, we use arguments to relate system- and component-level dependability attributes [4], and we apply uncertainty quantification (UQ) techniques focusing on the attributes of the LECs.

As its main contributions, this paper: (i) characterizes LEC assurance by identifying the relevant dependability attributes, specifying *assurance properties*, and developing a notion of *assurance measure* (see Section III-A); and (ii) provides a practical grounding to illustrate the feasibility of assurance quantification using an aviation system example: an autonomous taxiing capability for an unmanned aircraft system (UAS), focusing on the application of LECs as sensors in the perception function. Specifically, we elaborate one solution to quantify LEC assurance, based on Bayesian non-parametric UQ techniques, in particular Gaussian process (GP) regression.

We additionally discuss the relevance and contribution of component-level assurance to system-level assurance within a wider framework of DACs.

II. Related Work

Our prior work on AC confidence quantification [8] is closely related to the work in this paper. Where that work sought to build the quantification model based on the structure of the underlying argument, the current work proposes to use arguments mainly for the decomposition and refinement of assurance properties. Also, for assurance quantification we use Bayesian non-parametric UQ techniques. Other literature on confidence quantification for assurance purposes, e.g., [9], has investigated the application of evidential theory, although it has not been applied to the assurance of LECs.

Our original concept of dynamic safety case (DSC) [7], in which assurance quantification was a core principle, has since been applied for engendering trust in adaptive software [10]. Although that work has not considered LECs, it leverages probabilistic model-checking techniques, which can provide a quantitative notion of assurance (akin to what we will develop later, in Section IV), if LECs can be represented using the state-space models that are compatible with the verification techniques, e.g., as in [11]. Nevertheless, the core aspect of assurance in that work is an updatable assurance argument, rather than a quantified assurance measure.

Analogous to our notion of DSCs, dynamic safety management [12] is a proposed run-time methodology for assurance given in terms of quantified safety risk. However, this plausible theoretical basis for quantification is largely speculative about the applicability to LECs.

Quantification of application independent LEC (assurance) properties has also been explored in the recent literature, e.g., using Bayesian neural networks (NNs) to quantify model uncertainty [13], and using GPs to quantify robustness against adversarial examples and input perturbations [14]. This is compatible with our methodology (Section III-C), although what differentiates our work is that our approach links such quantified LEC properties to component and system-level dependability attributes, thereby giving a domain-specific semantics to quantification, in terms of assurance.

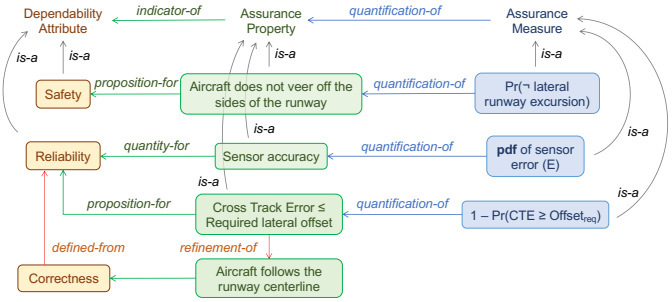


Fig. 1. Assurance properties and assurance measures: concepts facilitating the characterization of assurance. The properties and measures shown relate to our illustrative example (Section IV).

III. Solution Overview

A. Concepts and Related Terminology

Assurance is (the provision of) *justified confidence* that a system (or component/item, or service) possesses the applicable dependability attributes.

An *assurance property* is a (logical or probabilistic) proposition, or a quantity, associated with a dependability attribute. For instance (Fig. 1), sensor accuracy is a component-level assurance property that is also a quantity associated with the dependability attribute of *reliability*. On the other hand, the proposition that (aircraft) *cross track error* (CTE) is within the bounds of a required lateral offset is a *system-level* reliability assurance property. That, in turn, is a refinement of a *correctness*[3] assurance property, i.e., the proposition that the aircraft follows the runway centerline. Specifically, showing (over time) that the system correctly maintains CTE within the required lateral offset bounds provides assurance for reliability, refining the required correct behavior that the aircraft follows the runway centerline. Likewise, the proposition that the aircraft does not veer off the sides of a runway is a system-level *safety* assurance property.

An *assurance measure* is a suitable probabilistic quantification of an assurance property—i.e., both the interpretation and the associated numerical values that are assigned to the underlying quantity or proposition. Candidate assurance measures corresponding to the previously mentioned (reliability) assurance properties (Fig. 1) are, respectively, the probability density function (PDF) of the sensor error, and the probability that the CTE does not exceed the required lateral offset. The *uncertainty* in an assurance measure expresses the *confidence* associated with the assurance property, and thereby quantifies the extent of assurance.

In general, we can develop a plurality of assurance properties and assurance measures for a given dependability attribute. Attributes such as safety are largely meaningful in a system context, although we can formulate assurance properties (both at a system- and a component-level) that are related to other dependability attributes, and that have a bearing on safety. For

[3]We include correctness as an attribute to admit a formal methods concept for software, and relate it to the more general dependability attribute of reliability.

example, when a component is part of a safety system, its reliability (correctness) directly contributes to the delivery of the safety function and, thereby, to system safety.

From Fig. 1 we can intuit that—based on the dependability attributes being considered—assurance properties correspond to the subset of both functional and non-functional requirements that can be reasonably quantified via assurance measures. As we will see subsequently (Section III-C), assurance properties for a system and its constituent components are themselves interrelated. Next, we characterize the scope of LEC assurance as context for our approach to quantification (and its generalizability).

B. Problem Scope

There are number of choices to make when using LECs to implement the high level functions of perception, localization, planning, and control, in an autonomous vehicle architecture. For instance, we can either implement each function in the pipeline, individually, as an LEC; alternatively, a combination of functions (e.g., perception and localization) may be candidates for LEC-based implementation. There are also so-called *end-to-end* implementations in which a single LEC maps sensor inputs to low-level control outputs [15].

Next, an extremely rich variety of learning schemes, algorithms, and models are available to construct LECs suitable to the requirements, e.g., perception and localization functions have been implemented as (deep) convolutional neural networks (CNNs) trained using supervised learning. Based on the sensors available, such deep CNNs can be used not only for detection, classification, and tracking of various entities in the operating environment (broadly, classification problems), but also for predicting continuous quantities such as distance, speed, heading, etc. (more generally, regression problems). Deep CNNs have also been shown to be applicable for implementing the control function [15], while alternative implementations for planning and control functionality can use DNNs (which may or may not include convolutional layers) trained using reinforcement learning (RL) [16].

The preceding options represent only a small selection from the space of possible LEC implementation choices, and the literature is replete with other novel combinations of learning models and schemes [17]. As such, LECs pose particular assurance challenges that are, largely, specific to the algorithms used, implementation choices made, and their application and usage context. For instance, RL poses the risk of *reward hacking*: exhibiting unexpected emergent behavior wherein a policy is learned that attains the reward objectives at the expense of the (longer term) mission or safety objectives [18].

LECs are susceptible to *dataset shift*, i.e., encountering environments and data not representative of those used for training. Moreover, since LECs are expected to generalize to situations on which they need not have been trained, the errors produced may manifest as safety-critical consequences [19], when propagated through the pipeline from sensing to actuation (usually in combination with other system states and environmental conditions). Note that, in this context, *error* refers to the discrepancy between the LEC prediction (its output), and the revrequired or expected output. This error may or may not lead to a LEC *failure*, e.g., when the error exceeds a specified bound.

Next, we propose a generalized methodology for LEC assurance quantification; although, in light of the preceding discussion, we anticipate that concrete solutions—instances of our methodology—for providing (and quantifying) assurance are likely to be specialized to the particular application or usage of LECs, and the implementation choices made.

C. Methodology

Our overall approach to LEC assurance quantification involves: 1) characterizing assurance in terms of the dependability attributes, assurance properties, and assurance measures, taking into account the application/usage of the LECs and their implementation; and 2) applying uncertainty quantification (UQ) techniques to determine the numerical values to be assigned to the identified assurance measures.

1) Characterizing Assurance for LECs: Relating assurance to dependability attributes (as we have in the preceding narrative) broadly encompasses the compatible notion of assurance as the confidence that a system functions as intended in its usage environment. Here, two insights are noteworthy: first, as mentioned earlier (Section III-A), assurance properties can be seen as a subset of the appropriate system- and component-level requirements (in particular, those that can be reasonably quantified), so that the evidence that the requirements are satisfied contributes to increased assurance. This lends itself to the application of the entire suite of techniques for requirements decomposition and refinement, e.g., as in [20], in order to develop assurance properties and measures for a given LEC.

Second, we can map assurance properties to the claims made in an assurance argument, wherein a combination of inductive and deductive reasoning linking those claims to the evidence supplied, provides the basis to conclude that the claims have been met. For instance, in Fig. 1, we can state the system-level reliability and correctness assurance properties as claims in a structured argument that (i) provides the rationale for the *refinement-of* relation between the respective assurance properties; and (ii) further refines/decomposes the reliability assurance property into lower-level assurance properties, e.g., of sensor accuracy, that can be ultimately linked to the substantiating evidence items. Such arguments in themselves provide qualitative assurance, supplementing that obtained from quantifying the associated assurance measures. Thus, another approach for developing assurance properties and measures for a given LEC application context and is to create structured assurance arguments, for example, using the methodology in [21]. Note that the two approaches that we have mentioned here are themselves complementary.

2) Uncertainty Quantification: We can apply Bayesian parametric or non-parametric UQ techniques for assurance quantification, based upon the specific assurance measures being considered.

Authors' Preprint Copy. 15th European Dependable Computing Conference (EDCC 2019). Sep. 2019.
The definitive version of this article appears in the EDCC 2019 Conference Proceedings on IEEExplore.

a) Bayesian Parametric UQ: Here, a parametric probability distribution is assumed to represent the uncertainty associated with the LEC assurance measure of interest. If A is a random variable (RV) for the assurance measure, $\boldsymbol{\theta}$ are distribution parameters, and M_i represents a class of probability distributions selected from a set of available candidate distributions, then the probability distribution $A \sim \Pr(A|\boldsymbol{\theta}, M_i)$ characterizes the associated uncertainty. The idea, then, is to identify the parameters of the probability distribution in a Bayesian framework. To this end, the prior probability distribution of the parameters, $\Pr(\boldsymbol{\theta}|M_i)$ (representing our prior knowledge), is weighted by the likelihood of the parameters, $\Pr(\mathbf{E}|\boldsymbol{\theta}, M_i)$, based on observed *evidence*, $\mathbf{E}$. Normalizing the weighted prior using Bayes' theorem, we can determine the posterior distribution of the parameters, $\Pr(\boldsymbol{\theta}|\mathbf{E}, M_i)$, to establish not only the expected value of the parameters but also the associated uncertainty (e.g., in the model identification process). For more details, we refer the reader to [22].

b) Bayesian Non-parametric UQ: Where parametric UQ techniques are appropriate for a limited number of parameters, Bayesian non-parametric models relax the assumption of a particular parametric form for the uncertainty distribution. That is, rather than assuming a distribution with random parameters, the RV for the assurance measure is a random function over which a prior is assumed, and updated within a Bayesian framework. Effectively, a much larger (potentially infinite) parameter space is admitted, with greater flexibility in representing the nature of uncertainty, although there is a trade-off with respect to computational cost.

A Gaussian process (GP) [23] is a versatile prior, often used in non-parametric Bayesian UQ, representing a continuous stochastic process such that the joint distribution of the functions over which it is a prior, is (assumed to be) a multivariate Gaussian. A *covariance* function is a central concept to GPs, and it models the correlations between different points in the stochastic process. We adopt non-parametric Bayesian UQ using GPs as our approach to LEC assurance quantification and defer further discussion on the specifics to Section IV-D, where we illustrate its application in the context of a running example (discussed next).

IV. Illustrative Example

We now present an example drawn from the aviation domain to illustrate the application of our methodology for LEC assurance quantification. From the standpoint of LECs, the example highlights their usage as sensors in the perception function. More specifically, we consider deep CNNs trained offline in a supervised learning scheme for regression problems.

A. System Description

The target application is an autonomous taxiing capability to be deployed in an unmanned aircraft system (UAS). The overall goal is to facilitate the aircraft taxiing on an airport taxiway or runway without human pilot input, whilst meeting the applicable functional and safety objectives (described next in Section IV-B). To meet this goal, the perception function ingests video from a wing-mounted camera pointed towards the nose of the aircraft, producing output that is then processed by the localization, planning, and control functions, which, in turn, actuate and move the aircraft as required.

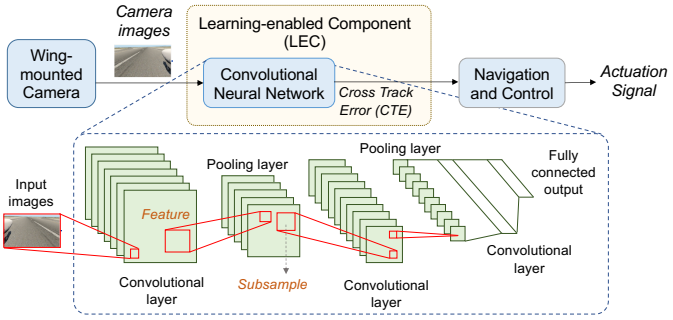


Fig. 2. Notional deep CNN implementing an LEC that processes camera images and estimates cross track error (CTE), as part of the perception function for an autonomous taxiing capability in an unmanned aircraft system.

Fig. 2 shows a simplification of this pipeline in which the perception function is an LEC that is implemented, notionally, as a deep CNN processing camera images as input. The input layer is (360×200) pixels $\times$ 3 channels wide; the network size and complexity is of the order of 100 layers with greater than two million tunable parameters. The LEC output of interest for this paper is a prediction of the *cross track error* (CTE)—in this application, it is distance between the aircraft nose wheel and the centerline of the taxiway or runway—together with the location of the nose wheel relative to the centerline. Effectively, the LEC performs regression under supervised learning to construct a mapping from a vector of real values (i.e., images) to a signed, real valued scalar (i.e., CTE). The absolute value gives the magnitude of the distance, and the sign indicates location, i.e., to the left $(-)$ or to the right $(+)$ of the centerline. The CNN output of CTE is then input to a traditional (i.e., non-LEC) implementation of the navigation and control function that interprets the input and generates the appropriate signals, e.g., thrust, steering, and flight control surface actuation (Fig. 2).

B. Scope of System-level Assurance

The core functional objective for taxiing, whether under direct control of a human pilot or autonomously, is to be aligned with, and to safely follow, the taxiway or runway centerline when taxiing, during takeoff, and after landing [24].

In general, alignment with the centerline requires that the roll (longitudinal) axis of the aircraft be parallel to the centerline and (possibly) offset on either side by no more than a specified threshold. From a system safety standpoint, among the main safety goals is to avoid *runway excursions*, i.e., veering off a runway or taxiway surface, either laterally, or by overshooting either end. For the autonomous taxiing capability in particular, these objectives are to be accomplished without pilot input, under a variety of diverse environmental situations, e.g., nominal weather conditions, in low visibility, at night time, in rainy conditions, under crosswinds, etc.

For each of the objectives above, we can formulate system-level assurance properties linked to the dependability attributes of interest. Then, following our methodology (Section III-C) we can specify the assurance properties/requirements) on the perception function, and on the LEC prediction of CTE in particular. These address both (i) the functional capabilities, e.g., providing CTE as a signed numerical value falling within a specified interval, and (ii) the non-functional attributes, e.g., providing reliable estimates of CTE (considered next).

C. Assurance Properties and Assurance Measures

We can further refine the objective of providing reliable estimates of CTE into reliability assurance properties, including: (i) the sensor is operating and functional when required, (ii) a normally operating sensor produces an accurate, precise, stable, and robust estimate[4] of CTE, and (iii) an inoperable sensor either produces no output, or produces an explicit error output. The latter two above together imply that there are no spurious outputs. Note that this is not a comprehensive enumeration of the applicable assurance properties for the LEC of Fig. 2, but a representative example. The LEC in our running example can be considered to be a (software) sensor producing a measurement—in fact, an inference [25]—on position and location. Henceforth, we will refer to the LEC of the running example as the *sensor*.

There are generic, measurement-based definitions for accuracy, precision, stability, and robustness [26], which we now specialize for the sensor. *Accuracy* is the *closeness* of the LEC estimate of CTE to a reference (true) value (also known as the *ground truth*) evaluated over all inputs. Accuracy is inversely proportional to the *sensor error*: the difference between the estimated CTE and the true value that is known during training and in validation, but may not be known in testing and operation. The greater the accuracy, the lower the sensor error. *Precision* is the amount of variability in repeated estimates of CTE, for a given input. In operation or testing, a CNN that has been trained in an offline supervised learning scheme (as in our running example) has fixed parameter values. Thus, for a given image, the output is fixed (i.e., it is deterministic). In other words, the sensor response is not expected to change when a given image is repeatedly supplied as input. As such, the quantity of precision does not have a meaningful interpretation for a single image.

Stability represents the amount of drift in CTE estimates, for a sequence of samples. For the LEC of the running example, quantifying stability involves determining the variation in the sensor error over a sequence of images that represent the time series progression of the camera images recorded as the aircraft moves on the runway during taxiing, takeoff, or after landing. *Robustness* gives the amount of variability in the estimates of CTE in response to abnormal inputs.

Other types of assurance properties can include expressions involving thresholds applied to the quantities indicated above, e.g., that accuracy exceeds some minimum level. The assurance measure for each of the above assurance properties is the respective probability distribution. For example, consider LEC accuracy for our running example: a useful assurance measure for this component-level (reliability) assurance property is the PDF of the sensor error (Fig. 1), whose expected value is the most likely numerical value for sensor error, and whose variance gives a characterization of the uncertainty (and, in turn, the confidence, i.e., assurance) in the same.

[4]The quantities of accuracy, precision, stability, and robustness are, in fact, reliability assurance properties that can apply to distance/position sensors in general.

D. Quantifying LEC Assurance

To illustrate LEC assurance quantification in the context of our running example, we mainly focus on LEC accuracy (equivalently, sensor error). As previously mentioned, LEC precision is not meaningful in this context. We do not address (assurance of) stability, robustness, or other LEC assurance properties in this paper, due to space constraints.

1) Gaussian Process Modeling: We model the assurance measure for sensor accuracy as a stochastic error surface, using Bayesian non-parametric UQ techniques, in particular a Gaussian process (GP) regression model [23].

Let $[\mathbf{X}, \mathbf{E}]$ be the set of training data points, where $\mathbf{X}$ is a vector of sensor inputs, and $\mathbf{E}$ is the vector of corresponding sensor errors. Let $\mathbf{E'}$ be the GP prediction of sensor error for an input from the test data, $\mathbf{X'}$; then the joint probability distribution of the observed sensor errors and the GP prediction of sensor error is a multivariate Gaussian distribution:

$$\begin{bmatrix} \mathbf{E} \\ \mathbf{E'} \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) & K(\mathbf{X},\mathbf{X'}) \\ K(\mathbf{X'},\mathbf{X}) & K(\mathbf{X'},\mathbf{X'}) \end{bmatrix}\right) \tag{1}$$

$K(A, B)$ is a *covariance matrix*, whose elements are the values of a suitable *covariance function*. The latter is evaluated at all pairs of the training and test data, $(\mathbf{X}, \mathbf{X'})$, and models the correlation between those data. For the running example, we have used the *squared exponential* covariance function [23], which models the intuition that the data that are *closer* to each other are more correlated, while those that are farther away are less correlated. The posterior distribution of the GP prediction of sensor error is given from Bayesian update, as:

$$\begin{aligned} \mathbf{E'} \mid \mathbf{X}, \mathbf{X'}, \mathbf{E} \sim \mathcal{N}(&K(\mathbf{X'},\mathbf{X})K(\mathbf{X},\mathbf{X})^{-1}\mathbf{E}, \\ &K(\mathbf{X'},\mathbf{X'}) - K(\mathbf{X},\mathbf{X})^{-1}K(\mathbf{X},\mathbf{X'})) \end{aligned} \tag{2}$$

Note that other covariance functions may be chosen [23] to explore how different GP models perform, although that tradeoff is not in scope for this paper.

Fig. 3 shows how we build and train the GP represented by equations (1) and (2). We construct the training data for the GP model from samples of the training and validation data of the LEC. More specifically, we use the features from the first fully connected layer after the convolutional layers, and apply principal components analysis (PCA) to reduce the dimensionality of the data set, $[\mathbf{X}, \mathbf{X'}]$. Then we apply clustering techniques—such as *k-means clustering*—to select the samples supplied to the GP. For these samples, since the true value of CTE is known a priori, we can also compute

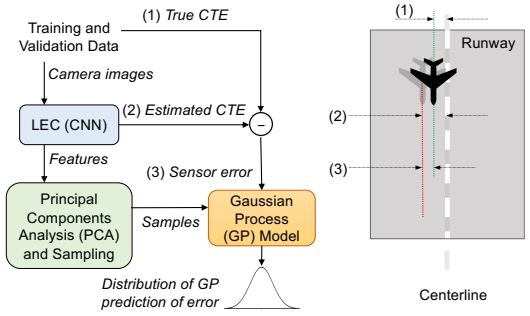


Fig. 3. Training a GP model to learn and predict sensor (LEC) error.

the sensor error, $\mathbf{E}$. For brevity, we omit other details of the GP learning and training process, as well as the details on its computational implementation.

2) Discussion: The idea is to model (learn) the sensor error performance and extrapolate the results to predict sensor error ($\mathbf{E}'$) in operation and testing. The extent to which the GP is trained induces uncertainty in its prediction of sensor error. This uncertainty is captured by the GP output: a Gaussian distribution whose expected value is the prediction of sensor error that is most likely, and whose variance is the uncertainty in that prediction. This distribution captures the full scope of uncertainty in the predicted sensor error for a single sample input image, i.e., the instantaneous assurance in LEC accuracy. Also, the posterior distribution is conditioned on *all* data on which it has been trained, including the input on which the prediction is made. This is in contrast to the Bayesian parametric approach, where the posterior distribution of error is conditioned *only* on the sensor error.

When the assurance property involves bounds on accuracy, i.e., that the sensor error falls within a certain range, or that it does not exceed some required maximum value, then by evaluating the cumulative distribution function (CDF) of the GP output at the bounds, we can compute the probability that the assurance property holds. In fact, since it is a posterior distribution computed from Bayesian update, the bounds represent the Bayesian *credible interval* and the probability represents *confidence*.

The rationale underlying our approach is that, as long as the GP model is itself shown to be accurate relative to the sensor:

– For the GP training inputs, the uncertainty in the predicted error is zero and there is full confidence in the sensor performance, i.e., the expected value of the GP output is the observed sensor (LEC) error, and the variance in this expectation is zero.
– For inputs drawn from the *training environment*—distinct from the GP training inputs—we can quantify the uncertainty in the predicted sensor error (which is governed by the covariance matrix/functions chosen). In turn, when the bounds on accuracy are known, we can gauge the confidence (assurance) in sensor accuracy as a probability. Note that, for our running example, we did not have an accuracy requirement, hence we characterize uncertainty, by providing the distribution of the predicted sensor error.
– On inputs outside the training environment, there can be an abnormal increase in the GP output uncertainty, where a decision can be made as to whether or not to trust the sensor (or even the GP model for that matter). Effectively, this can be a candidate runtime monitor for detecting dataset shift.

E. Experimental Results

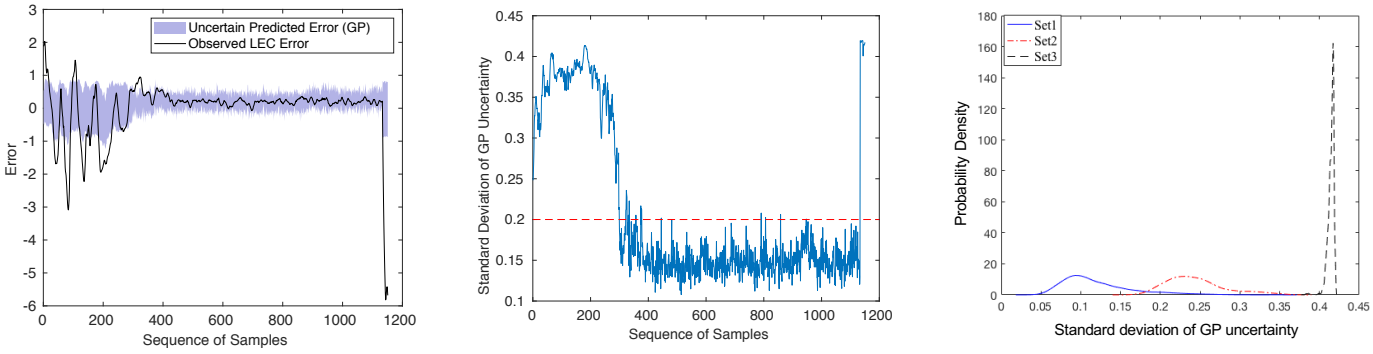
We now present some results of our experiments in quantitatively characterizing LEC assurance (i.e., LEC accuracy given in terms of the sensor error), based upon data generated from a simulation platform. In brief, we use a commercial-off-the-shelf flight simulator instrumented to reflect the pipeline architecture of Fig. 2. The simulation environment includes various airports from which we chose a specific airport having runway and taxiway surfaces with centerlines of varying quality (e.g., portions of the centerline may be obscured at various locations on the runways).

The simulator additionally allows changing various aspects of the simulation (to create different training and test environments), two of which we selected: (i) weather conditions: *Clear*, *Overcast*, *Foggy*, *Low visibility*, and *Scattered clouds*; and (ii) the time of day: *7am*, *Noon*, *2pm*, and *6pm*. For example, some environments are: *Clear Noon*, *Overcast Noon*, *Foggy Noon*, *Clear 6pm*, *Foggy 7am*, etc. For these, we gathered images via automated screen-capture (simulating the camera output) whilst taxiing the aircraft on the airport runway, not only using different software controllers, but also under manual control (via an external joystick and throttle). In parallel, for each image, we gathered actual CTE (from internal simulation variables), and LEC estimates of CTE (henceforth, referred to as *sensor outputs*).

The combination of images, corresponding actual CTE values (serving as ground truth), and sensor outputs comprise a data set. We used several data sets, one for each of the different environments identified above. Of these, samples drawn from the *Clear Noon* data set were used to train the LEC. That is, the *LEC training data* comprises samples of the pair of input image and corresponding actual CTE, drawn from the data set representing the *Clear Noon* training environment.

Then, we ran the trained LEC on 5600 samples drawn from data sets representing the *Overcast Noon* test environment as well as the *Clear Noon* training environment, to obtain the sensor output (and from it, the sensor error). From this collection of data, we drew 600 samples (as described earlier in Section IV-D1, and Fig. 3) to serve as the *GP training data*. In other words, the GP training data comprises samples from the LEC training *and* LEC test environments. Finally, to test the GP we used the following data sets, samples from which were unseen by the GP: *Set1*: *Clear Noon* and *Overcast Noon* (excluding the 600 samples that are the GP training data); *Set2*: *Foggy Noon*; and *Set3*: *Clear 6pm*.

For all data sets in our experimental setup, because we know the true CTE, we can compute the actual sensor error. Also, intuitively, since we expect the GP to perform well (accurately predict sensor error) in its training environment, to characterize the *quality* of the GP we compare its prediction of sensor



(a) (b) (c)

Fig. 4. Experimental results for quantifying LEC accuracy and assurance: (a) GP uncertainty in the predicted sensor error for unseen (test) data drawn from the *Set1* data set. For the first ≈ 350 samples, the GP has a large uncertainty, i.e., neither the GP nor the LEC can be trusted. For the remaining samples, the GP accurately predicts LEC error and the associated uncertainty (b) Standard deviation of GP uncertainty in the predicted sensor error, for the same sequence of samples as in Fig. 4a, showing an abnormal increase in the uncertainty for the first ≈ 350 samples (c) Probability density of the standard deviation of GP uncertainty in the predicted sensor error, for the *Set1*, *Set2* and *Set3* data sets. The density for *Set1* represents the uncertainty in the predicted sensor error in the training environment; the densities for *Set2* and *Set3* are noticeably shifted, indicating both dataset shift and increased uncertainty in the GP prediction.

error against the actual sensor error for the data samples from *Set1*. Since the GP output is a Gaussian distribution (with different mean and variance for each test input), we determine the fraction of all sensor errors from *Set1* that fall within 1σ and 2σ around the mean value of the GP output. These are, respectively, 68% and 91%. Note that both σ (standard deviation) and the mean value are predicted as parameters of the GP output. Since the values within 1σ and 2σ around the mean value account for about 68% and 95% of normally distributed data, the GP can be considered to be reasonably accurate in its training environment.

Fig. 4a visually illustrates this, plotting (i) the area within 2σ around the expected value of the GP prediction of sensor error, against (ii) the actual value of sensor error, for a *Clear Noon* data set, where the aircraft was controlled by a software controller. From the figure, we see that the GP accurately predicts LEC error for the sequence of samples 350–1200. We additionally observe (from Figs. 4a and 4b) that for the first ≈ 350 samples, the GP uncertainty in the predicted sensor error is not only large, but also the 2σ bound does not include the actual sensor error for some samples.

To diagnose this observation, we plot (in Fig. 4c) the distribution of the standard deviation of the the GP uncertainty in its prediction of sensor error, for all three data sets, i.e., the unseen test set (*Set1*) drawn from the GP training environment; and the tests sets from the unseen environments, *Set2* and *Set3*. Fig. 4c indicates that the distributions are, evidently, different. We can view this as an indicator of data distribution shift. That is, in the unseen environments, neither the GP nor the LEC can be trusted. Since the distribution of *Set1* is from the training environment where the GP is known to perform reasonably well, we show the 2σ bound of that distribution (as shown by the dotted red line in Fig. 4b) to discriminate an abnormal increase in uncertainty. As such (in Fig. 4b), for the first ≈ 350 samples from the *Set1* data set, we hypothesize that the increase in uncertainty could be attributed to insufficient sampling of the training environment whilst creating the GP training data. This simple indicator of dataset shift deserves a more in-depth study, which we will investigate as future work.

V. Concluding Remarks

So far as we are aware, this work is the first to connect dependability attributes, UQ, and their applicability to LEC assurance. Conservatively, we expect that our approach is generalizable to those LECs trained in an offline, supervised learning scheme, addressing regression and classification problems where there is a reference or *true* value against which LEC output can be compared. In this paper, we have shown the applicability to a regression problem using a worked aviation-related example. Exploring how our approach applies to reinforcement learning (RL), is an avenue for further research.

By using GPs for UQ, our approach implicitly assumes that the training and test data belong to a joint multivariate Gaussian distribution. Additionally, we have assumed a specific form for the covariance function (squared-exponential) in our example, although others can be chosen. Similarly, we can choose other non-parametric priors instead of GPs. In this paper, we have neither explored the suitability nor compared the performance of using alternative covariance functions or non-parametric priors. This—and more generally, comparing the suitability of different UQ techniques for assurance quantification—is another line of future work.

GPs, and Bayesian non-parametric UQ approaches are generally more flexible than Bayesian parametric techniques, but command a larger computational cost. GPs are known to scale poorly to high-dimensional data. Indeed, recall that we train the GP on the features that the LEC extracts (Section IV-D). Thus, problems affecting feature extraction—e.g., feature similarity for markedly different images/inputs—and the possibility of the PCA excluding features that potentially have a large impact, can affect GP training and, thereby, its predictive performance. Similar to LECs, the selection of the GP training samples also affects its predictive performance. A third avenue for future work is, therefore, exploring techniques

to address both the dimensionality problem of GPs, and the selection of GP training data. Promising results about the equivalence between deep neural networks (DNNs) and GPs [27], and so-called *sparse* GPs [28], may provide a way forward. Another candidate line of future work is to investigate how potential alternative approaches to UQ (such as Bayesian NNs, and training LECs with *dropout* [13], which quantify uncertainty in the LEC output and not directly in assurance measures) may augment our proposed approach, e.g., through *ensemble* techniques.

Although component- and system-level assurance can be related through a chain of linked assurance properties and measures—e.g., in an assurance argument—relating the quantification approaches is an important avenue of future work. Put in the context of our running example, this will seek, for instance, to establish how GP uncertainty in the predicted sensor error relates to system-level assurance, given as the probability that the actual CTE does not exceed a required lateral offset threshold during taxi operations.

We have put forth a notion of assurance for learning-enabled components (LECs) based on identifying and relating the relevant dependability attributes to application-specific assurance properties and measures. Our approach to assurance quantification is to give a probabilistic characterization of assurance measures, towards capturing uncertainty (and, thereby, confidence) e.g., using Bayesian non-parametric techniques for uncertainty quantification. To illustrate (and practically ground) our approach, we have applied it to an aviation system example. In particular, using GP regression, we have quantified a specific assurance property (accuracy) for an LEC in its use as a sensor. We have also presented the results from simulation experiments, discussing the assumptions made, and the challenges that remain. This paper has focused on LEC assurance at the component-level. Ultimately, however, *system-level* assurance—of the learning-enabled system (LES) that contains the LECs—is what matters. We are currently developing techniques to relate the two, as part of a wider dynamic assurance case (DAC) framework that will incorporate other sources of assurance (e.g., classical runtime verification) towards a broader goal of dynamic assurance and, eventually, certification.

References

[1] S. Lin, Y. Zhang, C. Hsu, M. Skach, M.. Haque, L. Tang, and J. Mars, "The architectural implications of autonomous driving: constraints and acceleration," in *Proc. 23rd Intl. Conf. Arch. Support for Prog. Lang. and Oper. Syst.*, Mar. 2018, pp. 751–766.

[2] MITRE, "Perspectives on research in artificial intelligence and artificial general intelligence relevant to DoD," Office of the Assistant Secretary of Defense for Research and Engineering (ASDR&E), Tech. Rep. JSR-16-Task-003, Jan. 2017.

[3] B. Lascara, T. Spencer, M. DeGarmo, A. Lacher, D. Maroney, and M. Guterres, "Urban air mobility landscape report - Initial examination of a new air transportation system," MITRE CAASD, Tech. Rep. 18-0154-4, Apr. 2018.

[4] A. Avizienis, J. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable and Secure Comp.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

[5] R. Clothier, E. Denney, and G. Pai, "Making a risk informed safety case for small unmanned aircraft system operations," in *Proc. 17th AIAA Aviation Technology, Integration, and Operations Conf., AIAA Aviation Forum*, no. (AIAA 2017-3275), Jun. 2017.

[6] S. Burton, L. Gauerhof, and C. Heinzemann, "Making the case for safety of machine learning in highly automated driving," in *Comp. Safety, Reliability, and Security (SAFECOMP 2017)*, S. Tonetta, E. Schoitsch, and F. Bitsch, Eds. Sep. 2017, pp. 5–16.

[7] E. Denney, I. Habli, and G. Pai, "Dynamic safety cases for through-life safety assurance," in *2015 IEEE/ACM 37th IEEE Intl. Conf. Soft. Eng.*, vol. 2, May 2015, pp. 587–590.

[8] E. Denney, G. Pai, and I. Habli, "Towards measurement of confidence in safety cases," in *Proc. 5th Intl. Symp. Empirical Soft. Eng. and Measurement*, Sept. 2011, pp. 380–383.

[9] R. Wang, J. Guiochet, G. Motet, and W. Schön, "Safety case confidence propagation based on dempster–shafer theory," *Intl. Jrnl. Approx. Reasoning*, vol. 107, pp. 46 – 64, 2019.

[10] R. Calinescu, D. Weyns, S. Gerasimou, M. Iftikhar, I. Habli, and T. Kelly, "Engineering trustworthy self-adaptive software with dynamic assurance cases," *IEEE Trans. Soft. Eng.*, vol. 44, no. 11, pp. 1039–1069, Nov. 2018.

[11] R. Ivanov, J. Weimer, R. Alur, G. Pappas, and I. Lee, "Verisig: verifying safety properties of hybrid systems with neural network controllers," in *Proc. 22nd ACM Intl. Conf. Hybrid Sys.: Computation and Control*, (HSCC '19), 2019, pp. 169–178.

[12] M. Trapp, D. Schneider, and G. Weiss, "Towards safety-awareness and dynamic safety management," in *2018 14th European Dependable Computing Conf. (EDCC)*, Sep. 2018, pp. 107–111.

[13] R. Michelmore, M. Kwiatkowska, and Y. Gal, "Evaluating uncertainty quantification in end-to-end autonomous driving control," Computing Research Repository (CoRR) arXiv: 1811.06817 [cs.LG], Nov. 2018.

[14] L. Cardelli, M. Kwiatkowska, L. Laurenti, and A. Patane, "Robustness guarantees for bayesian inference with gaussian processes," in *Proc. 33rd AAAI Conf. on Artificial Intelligence (AAAI-19)*, 2019.

[15] M. Bojarski, D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," Computing Research Repository (CoRR) arXiv: 1604.07316 [cs.CV], Apr. 2016.

[16] P. Long, W. Liu, and J. Pan, "Deep-learned collision avoidance policy for distributed multiagent navigation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 656–663, Apr. 2017.

[17] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in *2017 IEEE Conf. Comp. Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 3530–3538.

[18] D. Amodei, C. Olah, J. Steinhardt, P. Christiano, J. Schulman, and D. Mané, "Concrete problems in AI safety," Computing Research Repository (CoRR) arXiv:1606.06565 [cs.AI], Jul. 2016.

[19] National Transportation Safety Board, "Collision between a car operating with automated vehicle control systems and a tractor-semitrailer truck near Williston, Florida, May 7, 2016," Highway Accident Report NTSB/HAR-17/02, Sep. 2017.

[20] A. van Lamsweerde, "Goal-oriented requirements engineering: a guided tour," in *Proc. 5th IEEE Intl. Symp. Req. Eng.*, Aug. 2001, pp. 249–262.

[21] E. Denney and G. Pai, "Tool support for assurance case development," *Jrnl. Autom. Soft. Eng.*, vol. 25, no. 3, pp. 435–499, Sep. 2018.

[22] E. Asaadi and P. Heyns, "A Computational framework for Bayesian inference in plasticity models characterisation," *Computer Methods in Applied Mechanics and Engineering*, vol. 321, pp. 455–481, 2017.

[23] C. Rasmussen and C. Williams, *Gaussian processes for machine learning*. MIT Press, 2006.

[24] US Dept. of Transportation, FAA Flight Standards Service, "Airplane flying handbook," FAA-H-8038-3B, April 2016.

[25] W. Estler, "Measurement as inference: fundamental ideas," *CIRP Annals*, vol. 48, no. 2, pp. 611 – 631, 1999.

[26] Joint Committee for Guides in Metrology, Working Group 2, *JCGM 200:2012 International vocabulary of metrology – basic and general concepts and associated terms*, 3rd ed., BIPM, 2012.

[27] J. Lee, J. Sohl-Dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri, "Deep neural networks as gaussian processes," in *Intl. Conf. on Learning Representations*, Apr. 2018.

[28] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," in *Advances in Neural Information Processing Systems (NIPS) 18*, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds., MIT Press, 2006, pp. 1257–1264.