Uncertainty-aware Prediction Validator in Deep Learning Models for Cyber-physical System Data

FERHAT OZGUR CATAK, University of Stavanger, Simula Research Laboratory, Norway
TAO YUE and SHAUKAT ALI, Simula Research Laboratory, Norway

The use of Deep learning in Cyber-Physical Systems (CPSs) is gaining popularity due to its ability to bring intelligence to CPS behaviors. However, both CPSs and deep learning have inherent uncertainty. Such uncertainty, if not handled adequately, can lead to unsafe CPS behavior. The first step toward addressing such uncertainty in deep learning is to quantify uncertainty. Hence, we propose a novel method called NIRVANA (uNcertaInty pRediction ValidAtor iN Ai) for prediction validation based on uncertainty metrics. To this end, we first employ prediction-time Dropout-based Neural Networks to quantify uncertainty in deep learning models applied to CPS data. Second, such quantified uncertainty is taken as the input to predict wrong labels using a support vector machine, with the aim of building a highly discriminating prediction validator model with uncertainty values. In addition, we investigated the relationship between uncertainty quantification and prediction performance and conducted experiments to obtain optimal dropout ratios. We conducted all the experiments with four real-world CPS datasets. Results show that uncertainty quantification is negatively correlated to prediction performance of a deep learning model of CPS data. Also, our dropout ratio adjustment approach is effective in reducing uncertainty of correct predictions while increasing uncertainty of wrong predictions.

CCS Concepts: • **Computing methodologies** → **Uncertainty quantification**; **Machine learning**; • **Computer systems organization** → **Embedded and cyber-physical systems;**

Additional Key Words and Phrases: Uncertainty, prediction validation, deep learning, Cyber-physical Systems

ACM Reference format:
Ferhat Ozgur Catak, Tao Yue, and Shaukat Ali. 2022. Uncertainty-aware Prediction Validator in Deep Learning Models for Cyber-physical System Data. *ACM Trans. Softw. Eng. Methodol.* 31, 4, Article 79 (July 2022), 31 pages.
https://doi.org/10.1145/3527451

1 INTRODUCTION

Deep learning (DL) is increasingly applied in different phases of the development lifecycle of **Cyber-Physical Systems (CPSs)**. Such CPSs have critical applications such as water treatment and distribution networks, smart power grids, robotics, autonomous vehicles, and medical

All authors contributed equally to this research.
The work is supported by the Co-evolver project (No. 286898/F20) funded by the Research Council of Norway under the FRIPRO program.
Authors' addresses: F. O. Catak, University of Stavanger, Simula Research Laboratory, Stavanger, Norway; email: f.ozgur.catak@uis.no; T. Yue and S. Ali, Simula Research Laboratory, Oslo, Norway; emails: {tao, shaukat}@simula.no.

1049-331X/2022/07-ART79
https://doi.org/10.1145/3527451

infrastructures [21, 55]. Both CPSs and DL have inherent uncertainty that could potentially lead to unreliable (e.g., unsafe) behaviors of CPSs if such uncertainty is not properly dealt with. This issue has been well understood by the community. For example, the survey conducted by Wickramasinghe et al. [58] concludes that a lot of DL solutions for CPS data just produce labels for input data and do not quantify uncertainty in their predictions, which consequently could lead to the lack of confidence in these solutions and even further can cause unpredictable and unsafe CPS behaviors. Therefore, DL models employed for CPS data shall be accompanied with an uncertainty quantification framework to quantify the uncertainty inherent in these models, followed by developing mechanisms to deal with such uncertainty.

Deep neural networks are black-box models due to their multilayered nonlinear structures, which have been criticized for being non-transparent and their predictions not identifiable by humans [5]. In the CPS context, such black-box **Machine Learning (ML)** models have been used to make critical predictions [3]. Existing **explainable artificial intelligence (XAI)** methods (e.g., [22, 42]) have been applied to ML models, which are based on image and text datasets to understand decisions made by such models [50] and assess the reliability of the decisions. However, applying XAI is not straightforward in CPS since DL models are deployed as an integrated part of the CPS, and it is often not possible to interfere with it during its operation. In addition, it is difficult to reason decisions made by CPSs deployed with DL models simply based on sensor data.

In this article, we study quantification of model uncertainty based on **Monte-Carlo Dropout (MC Dropout)** neural networks in prediction models developed from CPS data. MC Dropout was originally proposed by Gal [14] for uncertainty quantification and has been used for quantifying uncertainty in autonomous driving [39], robotics [41], and medical imaging [9]. We apply it also for uncertainty quantification, but our aim is to predict wrong labeling based on uncertainty quantification, which is important to prevent unsafe behaviors of CPS. In contrast to our work, existing studies, e.g., [26, 33, 49], focus only on quantifying uncertainty. We used the entropy and softmax prediction probability variance metrics to quantify the prediction time uncertainty of a DL model. Then, we use quantified uncertainties as a training set to create an additional ML model and predict incorrect labeling (*Prediction Validator* of the original DL model using **Support Vector Machine (SVM)** together with **Radial Basis Function (RBF)**).

For evaluation, we used the following CPS datasets: (1) the **Secure Water Treatment (SWaT)** available at [19], (2) Sensor readings from a wall-following robot (shortly, Robot) available at [13], (3) Video conferencing system made available at the GitHub repository,[1] and (4) KITTI available at [38]. Our results show that quantified uncertainties in the prediction time of each model constructed for each CPS dataset are strongly correlated with the prediction performance of the model. In order to investigate the correlation between uncertainty and model performance in the prediction time, we first quantified model uncertainty of all the models trained for all four datasets. Second, we sorted the test dataset according to uncertainty values in the descending form. To show the negative effects of the highly uncertain data on the DL models employed for CPS data, we removed the highly uncertain instances from the test dataset one by one. Hence, the remaining data's uncertainty values are lower than the whole test dataset. We repeated this experiment, leaving only 40% of the data in the test set (i.e., deleting 60%) step by step. Based on the validation results, we improved the models' F_1 metrics from 0.92 to 1.00 for SWaT, 0.89 to 1.0 for Robot, 0.79 to 0.97 for the Video conferencing, and 0.96 to 1.00 for KITTI datasets.

[1] https://github.com/unc-cps/dsn2021, password: 2020pwd.

Our research has made the following contributions:

- We demonstrated how uncertainties in the model's prediction time can be quantified using the entropy and softmax prediction probability variance.
- We demonstrated the correlation between model uncertainty of DL models and their prediction performance.
- We proposed "prediction validator," an SVM model with RBF kernel for validating prediction of DL models employed for CPS data based on quantified uncertainties. The most significant difference from the literature is that our prediction validator is a model, not just a threshold value.
- We iteratively re-trained DL models with highly uncertain inputs and improved their prediction performance.
- We conducted experiments with four real-world CPS datasets and proposed a mechanism to optimize dropout ratios.

The rest of the article is organized as follows: Section 2 introduces the related work. Section 3 presents the context of this work and also presents a running example. Section 4 describes preliminary information about model uncertainty and uncertainty quantification. Section 5 shows our system overview. Section 6 evaluates the proposed uncertainty quantification method. Section 7 shows the practical applications of NIRVANA. Section 8 concludes this article.

2 RELATED WORK

In this section, we discuss the related work from the following aspects: uncertainty quantification in DL (Section 2.1), uncertainty in software engineering (Section 2.2), and ML and CPS testing (Section 2.3), respectively.

2.1 Uncertainty Quantification in DL

Uncertainty quantification in DL models is an ongoing topic of intensive research. The main method for quantifying uncertainty in DL models is the use of the softmax variance and expected entropy over multiple models. The work by Gal [14] demonstrated that a neural network model with prediction-time-activated dropout is equal to a specific variational inference on a Bayesian neural network model. The prediction model uncertainty is approximated by averaging probabilistic feed-forward MC dropout sampling during the prediction time. This approach is highly effective in terms of practical implementation for large models and has proved useful in understanding the dynamics of the networks under different testing conditions [53].

In addition to MC dropout-based Bayesian Neural Network [36], an alternative uncertainty quantification method in DL is Deep Ensembles, a non-Bayesian method for uncertainty quantification [31]. In Deep Ensemble, several models are trained in parallel, and each of them is trained using random noise (with adversarial instances) in its subset part of the training dataset. The final output of the training phase is a set of independent classifiers, each of which has its own unique set of weights. The ensemble can predict the same input instance with different predictions using its individual models. Considering that Deep Ensemble requires several models, it is not suitable for single-model uncertainty quantification. Therefore, in this work, we opted for the prediction-time-activated dropout-based neural networks approach for single-model uncertainty quantification.

Methods in adversarial ML aim to create perturbed instances to evade (or fool) a DL model. Recently, Ma et al. [36] showed the correlation between uncertainty and model prediction performance. In their approach, to increase the DL models' robustness, they applied adversarial ML attacks (e.g., Fast Gradient Sign Method, Basic Iterative Method, DeepFool, Jacobian-Based Saliency Map Attack, and Carlini-Wagner) to create highly uncertain and misclassified instances.

Newly created adversarial instances are used as a new training dataset in the re-training phase to increase the robustness of the DL model. The iterative training with adversarial instances is like traditional adversarial training [20]. They used MNIST, Fashion MNIST, and CIFAR-10 image datasets in their experiments to evaluate the effectiveness of their approach. Though this method is similar to adversarial training, NIRVANA improves the prediction performance of DL models by using highly uncertain inputs instead of adversarial inputs.

Ghoshal and Tucker [18] studied how Bayesian **Convolutional Neural Networks (CNNs)** can quantify uncertainty in DL models to enhance the diagnostic performance of the human-machine alliance using a COVID-19 chest X-ray dataset. Another study reported in [45] uses uncertainty values and quality estimates to model predictive uncertainty estimation. Their main idea relies on the difference between the detection box and ground truth in images, and therefore is only applicable for object detection problems but not CPS data.

In our recent work [7], we proposed an uncertainty quantification metric for object detection models, named PURE. Experiments were performed on the KITTI, Stanford cars, Berkeley DeepDrive, and NEXET datasets, together with YoLo, SSD300, and SSD512 object detection models, to quantify their prediction uncertainty with PURE. PURE is an uncertainty quantification method in object detection models for autonomous driving; on the other hand, NIRVANA aims to assess the reliability of DL models' prediction with another model and improve DL models' predictions with generated uncertainty quantification results.

2.2 Uncertainty in Software Engineering

In the last decade, uncertainty has attracted significant attention in software engineering, due to the fact that uncertainty is gradually being recognized as an inevitable characteristic of complex systems such as CPSs, due to their increasingly unpredictable, open, and networked operating environments [16]. There exists a number of solutions (e.g., [2, 8, 60, 61]) in the literature for modeling or specifying known uncertainty on requirements, design models, and so forth. There also exist approaches in software engineering that focus on refining uncertainty measurements from historical data (e.g., [59]), reasoning with Bayesian inference (e.g., [52]), and testing software systems under uncertainty (e.g., [6, 35]). To compare with NIRVANA, these works are generic solutions and do not particularly focus on software system with DL models deployed.

Same as for conventional software systems, software systems with DL models employed also face quality and reliability issues. Not handling them properly has disastrous consequences. To this end, Zhang [62] proposed an approach to generate highly uncertain test inputs using adversarial ML attacks and improve prediction performance of DL models. More specifically, the approach first uses genetic algorithms to generate adversarial instances to effectively penetrate DL defense techniques, therefore uncovering hidden defects in the DL models. Second, it re-trains the DL models with adversarial inputs. Tuna et al. [11] showed that adversarial inputs generated by adversarial ML attacks can fool DL models with low prediction uncertainty values. These re-training approaches based on adversarial inputs are different from NIRVANA, as NIRVANA improves the prediction performance of DL models by re-training them with highly uncertain training inputs.

In autonomous driving, Michelmore et al. [39] applied variation ratios, entropy, and mutual information-based uncertainty quantification methods to discover the relation between uncertainty and wrong prediction of steering angles of autonomous cars. They found that wrong angle decisions of an autonomous car have significantly higher uncertainty values than correct decisions. Based on this observation, they defined a threshold to determine crash risks. This is different from NIRVANA, as NIRVANA employs an SVM-based prediction model (instead of a threshold) for prediction validation.

When particularly considering CPSs with DL models employed, it is critical to verify accurate interactions between a DL model and the physical operating environment of the CPS with the DL model employed and confirm its safety and correct operation [32]. Along this line, Dreossi et al. [10] proposed a toolkit, named VERIFAI, which focuses on the simulation-based safety analysis of systems enabled with artificial intelligence and ML components. VERIFAI especially explores challenges with implementing formal methods to understand ML models and analyze system behavior under environmental uncertainty. On the other hand, NIRVANA is not a formal method tool. Instead, it aims to predict wrong DL models' predictions based on uncertainty values.

2.3 ML and CPS Testing

There is an increasing interest in ML and CPS testing, and new techniques and research have been conducted toward ML and CPS testing. A survey conducted by Sherin et al. [47] classifies testing methods in ML models. They showed a dramatic increase in the number of publications between 2015 and 2018. Most ML testing methods are black box (74%), and more than half (57%) of the proposed ML testing methods are about testing neural networks models.

Asadollah et al. [1] classified CPS testing into six categories: hardware testing, structural and computation testing, extra-functional properties testing, network testing, integration testing, and system testing. DL models are now becoming one of the main components of CPSs. Thus, the CPS testing should also include DL model testing. In another work [64], the authors stated that a CPS is a complex system with the following features: data driven, software defined, and self-government. They also said that uncertainty modeling for CPS testing is one of the challenges for future complex CPS testing methods.

Berend et al. [4] showed that DL models do not provide statistical guarantee and have limited capability in handling data that falls outside of the training dataset's distribution (i.e., **out-of-distribution (OOD)** data). They conducted a large-scale study on the impact of data distribution on state-of-the-art DL testing techniques. Their results show that existing OOD detection techniques could distinguish OOD data from newly generated test cases and a distribution-aware dataset tends to be more effective in robustness enhancement. Our prediction validator model predicts wrong predictions of DL models employed for CPS data using uncertainty measurements, which is different from the aim of their study: enhancing model robustness with OOD datasets.

Zhang et al. [63] investigated the capability of different uncertainty metrics (including prediction confidence score and variation ratio of original prediction) in differentiating **benign examples (BEs)** and **adversarial examples (AEs)** of deep learning models. They further investigated uncertainty patterns of BEs and AEs generated by existing adversarial ML techniques and observed that these BEs and AEs follow certain uncertainty patterns and some patterns were largely missed. Based on these observations, they proposed a technique to automatically generate diverse adversarial data and concluded that uncommon data generated with their method is hard to be defended by existing defense techniques. To compare with this work, we identify wrong predictions with uncertainty measurements instead of generating adversarial data. In our recent study [12], we showed that uncertainty in predictions using inputs manipulated by adversarial ML attacks could be quite low and show similar patterns, just as in BEs.

Weiss and Tonella [56] proposed a fail-safe DL-based system with a supervisor that monitors the DL uncertainty for any given input at runtime such as ignoring predictions for high-uncertainty inputs and running a safe fallback process by stopping a self-driving car at the side of the street, for instance. The supervisor has the capability to discard predictions for highly uncertain inputs above a specific threshold. The threshold setting is critical since a high threshold will fail to reject many false predictions, while a low threshold will result in an excessive number of false warnings.

The authors introduced a new metric for determining the appropriate threshold value to address this concern.

3 CONTEXT AND RUNNING EXAMPLE

This section presents the context of this work together with relevant challenges in Section 3.1, followed by a running example that will be used to explain various concepts throughout the article (Section 3.2).

3.1 Challenges and Context

Developing CPSs is challenging as compared to classical software due to the unique characteristics of CPSs, such as the interdisciplinary nature (e.g., involving software, hardware, physics), close interactions with their operating environment including humans, and the use of communication mediums for communicating among their components [16]. In addition, CPSs face uncertainties both during their development and throughout their operation, e.g., due to intrinsic uncertainty of the DL models employed, inherently unpredictable environment, unpredictable human behaviors, and unreliable network communications among CPS components [35, 60]. Such uncertainties make CPSs' development and operation even more challenging. Thus, to better understand uncertainties of CPS behaviors with the aim to enhance the dependability of the next generations of CPSs, develop new test cases, and so forth, data produced during their operation can be used for various analyses with DL models. To this end, the context of this work focuses on CPS data produced, e.g., during their operation.

The overall context of this article is shown in Figure 1. Since we focus on uncertainty in DL models employed on CPS labelled data, we expect that there is an *Execution and Labelling System* that interacts with the CPS of interest, obtains relevant CPS data, and labels them with a DL model. Examples of such labels include whether a security attack was detected by the CPS, and whether or not an execution of the CPS system failed. We would also like to acknowledge that such labeled data are commonly available and we use them in this article for the evaluation purpose. The *Labeled Data* is then the input for *Uncertainty Quantification*, which consequently produces another set of data collected for uncertainty metrics (i.e., *Uncertainty Measurements*). These data are then the input to another ML model, *Prediction Validator*, which makes predictions about wrong classifications of DL models employed for CPS data due to uncertainty. In our context, the motivation of having *Prediction Validator* is to predict the correctness of the DL models employed for CPS data's output by using the model's uncertainty values. In a typical DL model training, the training dataset $\mathcal{D}$ inevitably cannot include all possible input instances. Thus, any model trained with CPS data has no chance to reach the goal of discovering the best parameters (i.e., weights in neurons) that can successfully map all possible inputs to CPS outputs. This makes an inequality between the ground truth y and the prediction of a model for CPS data $\hat{y}$, called *model uncertainty*. One straightforward strategy to reduce the model uncertainty is to obtain more data, which is often not possible in practice. For example, Kendall and Gal [29] stated that given enough data, model uncertainty could be reduced to a negligible level. This is a hard task that cannot be achieved for any realistic applications. In this paper, our main aim is about using uncertainty quantification metrics to predict wrong predictions of DL models employed on CPS labeled data. We end up with a prediction validator model that indicates how much a given DL model predicts correctly and how much the actual value (i.e., ground truth) is deviated from the predictions.

3.2 Running Example

To explain various concepts in our method, we will use the running example of the Robot dataset [13]. The dataset is collected from a robot that moves itself in a room with the help of 24 different

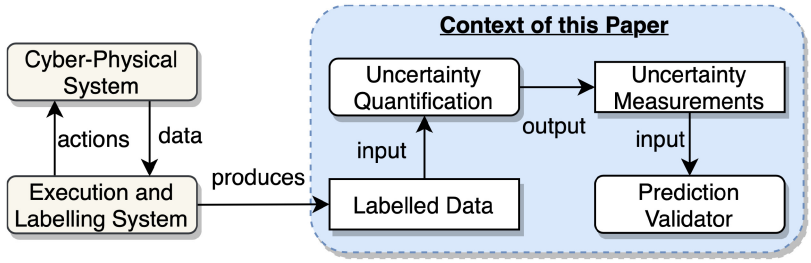


Fig. 1. Overall context.

Table 1. Sample Sensor Readings of the Robot Dataset

Row No	US1	US2	US3	US4	···	US22	US23	US24	Label
1	1.455	1.477	1.504	5.000	···	4.435	0.646	1.448	Move-Forward
2	1.111	1.140	2.331	2.331	···	1.079	1.064	1.128	Move-Forward
3	0.548	1.382	1.378	1.395	···	0.465	0.531	0.805	Sharp-Right-Turn
4	0.908	0.935	1.631	1.605	···	0.891	0.880	0.890	Move-Forward
5	1.273	2.418	2.420	2.628	···	0.854	0.862	4.328	Sharp-Right-Turn
6	1.407	1.444	1.458	2.622	···	1.404	1.384	1.392	Sharp-Right-Turn
7	0.806	1.486	2.744	2.756	···	0.759	0.765	0.789	Sharp-Right-Turn
8	1.272	1.518	2.423	2.423	···	1.196	1.181	1.206	Move-Forward
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$n-1$	1.089	1.680	1.671	1.681	···	1.304	1.089	1.077	Move-Forward
n	2.325	2.541	5.000	2.073	···	4.260	5.000	1.758	Sharp-Right-Turn

sensors attached circularly around its waist. Based on the current set of sensor values, the robot decides its next move, which is classified into four labels: Move-Forward, Slight-Right-Turn, Sharp-Right-Turn, and Slight-Left-Turn. Table 1 shows the excerpt of the labeled dataset. The columns *US1* to *US24* represent the 24 sensors. The sensor values are continuous. The *Label* column shows the label associated with each set of sensor values, i.e., each row in the table. To validate our method, we will build a DL model to predict the direction of motion based on sensor readings by using the dataset. We split the dataset as training dataset to build the DL model and test dataset to evaluate the DL model. We aim to quantify the uncertainty in the decisions of the DL model about making the next move.

4 PRELIMINARY INFORMATION

A typical ML classification model, h, has the following components: a training data $\mathcal{D}$, output labels Y, and a loss function ℓ. Given a dataset $\mathcal{D} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\} \subset X \times Y$, where $X \in \mathbb{R}^{m \times n}$ is the input instances, $\mathbf{x} \in \mathbb{R}^m$ is an m-dimensional input instance, and $Y \in \{1 \cdots C\}$ with C being the output labels, training instances $\{\mathbf{x}_i, y_i\}\, \forall X$ are a set of independent and identically distributed training instances from unknown probability measures P on $X \times Y$. We want to find a model $h : X \mapsto Y$ using weights of neural net parameters w and a loss function $\ell : Y \times Y \mapsto \mathbb{R}$ as similar as possible to the original function that has generated the labels.

The study by Gal [14] indicated that a Bayesian neural network model with prediction-time-activated dropout is comparable to a specific variational inference on a neural network model with prediction-time-activated dropout. Their approach is an ensemble approach. In each single

prediction, the system drops out different neurons in each of the network's layer according to the dropout ratio in the prediction time. The predictive mean is the average of the predictions over dropout iterations, T, and the predictive mean is used as the final prediction, $\hat{y}$, for the input instance $\mathbf{x}$. The overall prediction uncertainty is approximated by finding quantification metrics (e.g., the entropy and the variance of the probabilistic feed-forward MC dropout sampling) during prediction time. The final prediction is [14]

$$p(\hat{y} = c|\mathbf{x}, \mathcal{D}) \approx \hat{\mu}_{pred} = \frac{1}{T} \sum_{y \in T} p(\hat{y}|\theta, \mathcal{D}), \tag{1}$$

where θ is the model weights, $\mathcal{D}$ is the input dataset, T is the number of predictions of the MC dropouts, and $\mathbf{x}$ is the input sample. The label of input instance $\mathbf{x}$ can be estimated with the mean value of MC dropout predictions $p(\hat{y}|\theta, \mathcal{D})$, which will be done T times.

Random neurons in each layer are dropped out of the underlying neural network model during prediction time, according to the p, to build a new model. As a consequence, T distinct classification models may be employed to forecast the class label of the input instance and quantify the aggregate prediction's uncertainty.

For each testing input instance $\mathbf{x}$, the predicted label is assigned with the highest predictive mean. We used entropy and softmax variance measures in order to quantify how much the model is uncertain about its labeling predictions. The entropy is a measure defined as the average level of uncertainty fixed in the likely outcomes of a random variable [9]. The entropy function can be described as [46]

$$H(\hat{y}|\theta, \mathcal{D}) = -\sum_{c \in C} p\left(f(\mathbf{x}) = c\right) \log p\left(f(\mathbf{x}) = c\right), \tag{2}$$

where $f(\mathbf{x})$ is the classification model prediction, and $p\left(f(\mathbf{x}) = c\right)$ is the model's prediction probability, $f(\mathbf{x})$, for the class c.

The second approach is the softmax probability variance measure, which measures the uncertainty in the prediction as a function of the model's softmax probability values variance for each class. We calculate the T predictions' softmax output variance for each class and then average the results to find the mean variance over all classes. The softmax variance can be described as [23]

$$VAR\left(f(\mathbf{x})\right) = \frac{1}{T} \sum_{t=1}^{T} \left(f(\mathbf{x}) - f(X)\right)^2. \tag{3}$$

5 APPROACH

The main goal of this work is to find wrong labeling (i.e., predictions) of DL models for CPSs, based on model uncertainty quantification in the prediction time. The proposed uncertainty quantification approach mainly consists of three phases: (Phase I) the model building and retraining phase, (Phase II) the prediction-time-activated dropout-based neural-network-based predictions' uncertainty quantification phase, and (Phase III) the prediction validator model building phase (Figure 2).

As shown in Figure 2, Phase I is responsible for data pre-processing and model building with input dataset $\mathcal{D}$ from CPSs. A standard neural network training is performed at this phase, and both binary and categorical cross-entropy loss functions are used in weight optimization. The main outcome is the base model, which is the model with the best prediction accuracy and lowest loss over the entire training set. The most significant difference between the base model and a standard neural network model is that the prediction time dropout of the base model is active at the prediction time, while the standard neural network model has no dropouts. Here, we applied uncertainty-based iterative learning to improve the DL model's prediction accuracy. Accordingly,

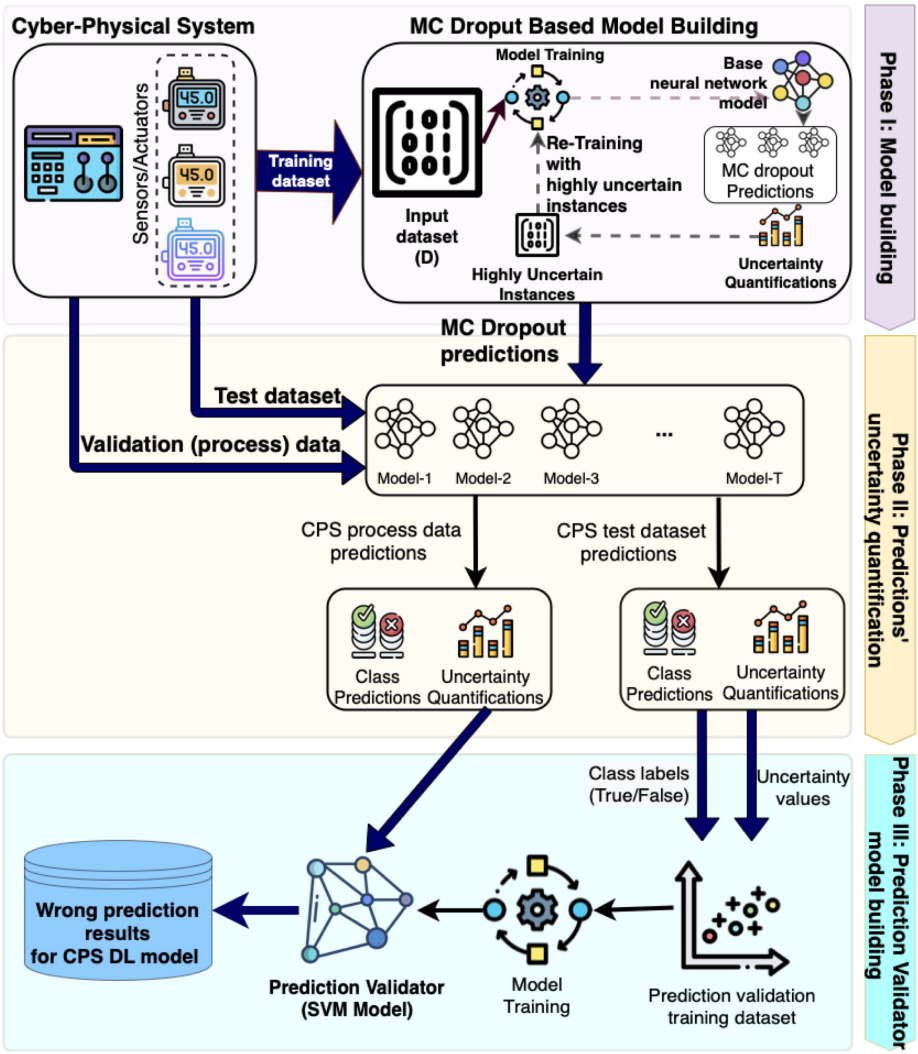


Fig. 2. Overview of NIRVANA.

(1) we train the DL model with a training dataset, (2) we quantify the uncertainty of the training set, (3) we rank the training dataset according to their uncertainty values, and (4) we re-train the DL model with the topmost highly uncertain instances to improve the prediction accuracy.

In Phase II, the base model deactivates several neurons in its every layer according to the dropout ratio p in each prediction T for an unseen input example $\mathbf{x}$. Figure 2 illustrates multiple predictions with T different models (*Model-1*, *Model-2*,... *Model-T*), which are generated from the base model. The selection of the dropout ratio will be discussed in Section 5.2. In this way, the base model can calculate the effect of the probability distribution of the weights on the MC dropout models' decisions based on the *softmax* output values, and the prediction uncertainty of the base model can be quantified with entropy values of the calculated prediction softmax distributions. Hence, the model's prediction uncertainty is directly related to its accuracy, which is the average of the softmax prediction accuracy. Another model uncertainty quantification metric

Table 2. Example Training Datasets: The Table on the Top Shows the CPS Data and the MC Dropout Model's Outputs; the Table Below Shows the Training Dataset for Prediction Validator

CPS Dataset					MC Dropout Predictions			
S_1	S_2	$\cdots$	S_n	**Real** (y)	**Pred.** ($\hat{y}$)	**Entropy**	**Variance**	$y == \hat{y}$
1.455	1.477	$\cdots$	1.448	Move-Forward	Move-Forward	0.560541	0.126364	True
1.111	1.140	$\cdots$	1.128	Move-Forward	Move-Forward	0.952304	0.304852	True
0.548	1.382	$\cdots$	0.805	Sharp-Right-Turn	Sharp-Right-Turn	0.612164	0.162749	True
0.908	0.935	$\cdots$	0.890	Move-Forward	Sharp-Right-Turn	0.988710	0.309782	False
				$\vdots$				
2.325	2.541	$\cdots$	1.758	Sharp-Right-Turn	Move-Forward	0.643164	0.189927	False

$\Downarrow$

Prediction Validator Training Dataset		
Entropy	**Variance**	$y == \hat{y}$
0.560541	0.126364	0
0.952304	0.304852	0
0.612164	0.162749	0
0.988710	0.309782	1
	$\vdots$	
0.643164	0.189927	1

is the softmax probability variance measure, which measures prediction uncertainty as a function of the model's softmax probability values variance for each class. As discussed in Section 4, we calculate the T predictions' softmax output variances for each class and then average the results to find the mean variance over all the classes.

In Phase III, class predictions and uncertainty quantifications are used as the training data to train the model for Prediction Validator, details of which will be discussed in Section 5.1. The output of Prediction Validator is wrong prediction results, which should be provided to relevant stakeholders such as CPS operators as an alarm. More discussions are provided in Section 7.

5.1 Prediction Validator

To predict wrong predictions of DL models employed for CPS data, we opted for an SVM-based detection model (i.e., prediction validator). We chose SVM since the scientific literature [28] showed that SVM's prediction performance is generally better than **multilayer perceptron (MLP)**. In the future, we will try with MLP-based classification models. The training dataset for the prediction validator is prediction outputs of the DL model employed for CPS data and their uncertainty (Figure 2). Wrong labeling information includes **false-positive (FP)** and **false-negative (FN)** labeled instances in the training dataset. **True-positive (TP)** and **true-negative (TN)** predictions of the DL model employed for CPS data are labeled as 0 in the prediction validator training dataset; FP and FN predictions are labeled as 1 in the dataset, as shown in Table 2.

Based on our running example of the Robot dataset in Section 3.2, we created a DL model to predict the direction of the motion. The DL model reads data from the test dataset as input values and then predicts the label. The training dataset might not contain enough instances to learn all situations. Our MC dropout-based approach will quantify uncertainty of the DL model's

Table 3. DL Model's Predictions with Uncertainty Values

Row No	Entropy	Pred. Variance	Prediction
1	0.560541	0.126364	Move-Forward
2	0.952304	0.304852	Move-Forward
3	0.612164	0.162749	Sharp-Right-Turn
4	0.988710	0.309782	Move-Forward
5	0.697368	0.225314	Sharp-Right-Turn
6	0.691121	0.222518	Sharp-Right-Turn
7	0.747086	0.244951	Move-Forward
8	0.643164	0.189927	Move-Forward
⋮	⋮	⋮	⋮
$n-1$	0.845818	0.275986	Move-Forward
n	1.258826	0.342335	Move-Forward

Table 4. Prediction Validator Results for the DL Model's Predictions

Row No	Entropy	Pred. Variance	Prediction	Actual	Prediction Validator
1	0.560541	0.126364	Move-Forward	Move-Forward	Correct
2	0.952304	0.304852	Move-Forward	Move-Forward	Correct
3	0.612164	0.162749	Sharp-Right-Turn	Sharp-Right-Turn	Correct
4	0.988710	0.309782	Move-Forward	Sharp-Right-Turn	Wrong
5	0.697368	0.225314	Sharp-Right-Turn	Sharp-Right-Turn	Correct
6	0.691121	0.222518	Sharp-Right-Turn	Move-Forward	Wrong
7	0.747086	0.244951	Move-Forward	Sharp-Right-Turn	Wrong
8	0.643164	0.189927	Move-Forward	Move-Forward	Correct
⋮	⋮	⋮	⋮	⋮	⋮
$n-1$	0.845818	0.275986	Move-Forward	Move-Forward	Correct
n	1.258826	0.342335	Move-Forward	Sharp-Right-Turn	Wrong

prediction with the entropy and softmax prediction probability variance. Table 3 shows the entropy and softmax prediction probability variance in the *Entropy* and *Pred. Variance* columns. These two values are associated with each row, i.e., one set of sensor values. The row numbers are the same as in Table 1; i.e., they identify the same set of sensor values. Higher entropy and prediction variance values indicate higher uncertainty associated with a prediction. For example, row 2 with the entropy value of 0.95 for the prediction *Move-Foward* has high uncertainty as compared with row 1 with the same prediction that has the entropy value of 0.56.

The prediction validator will validate the DL model's predictions, based on uncertainty values. Validation results are illustrated in Table 4. Wrong predictions are highlighted in the grey rows.

Suppose we created another dataset using the DL model employed for CPS and its uncertainty outputs shown in Figure 3. There are two class labels in the new dataset: correct and wrong predictions. The class labels are separable in the Euclidean space using a region-based classifier such as RBF kernel-based SVM. Then, the new classifier's output can be used to show whether the DL model employed for CPS data prediction for an input instance is likely correct or wrong. Let $X \in \mathbb{R}^{m \times n}$ denote the CPS test dataset and $\mathbf{y} \in \mathbb{R}^{m}$ denote the vector of corresponding labels for X;

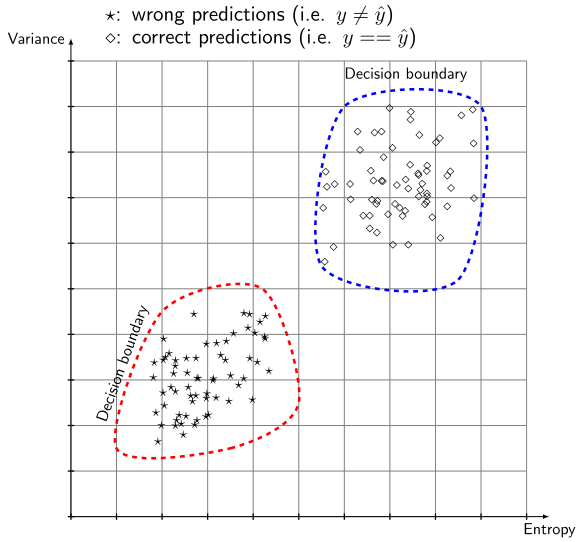


Fig. 3. Prediction validator model's uncertainty training dataset.

then the DL model employed for CPS data $h : X \mapsto Y$ outputs an uncertainty dataset $\mathcal{U} \in \mathbb{R}^{m \times k}$, where k is the number of uncertainty metrics (i.e., variance, entropy).

Algorithm 1 shows the uncertainty dataset generation and prediction validator model building. The DL model employed for CPS data is built in lines 1–2. The uncertainty dataset is created in lines 3–6. The prediction validator model is built in line 7.

ALGORITHM 1: Building prediction validator model

Input: Input CPS dataset: $(X \in \mathbb{R}^{m \times n}, \mathbf{y} \in \mathbb{R}^{m})$, uncertainty metrics list: K

// Create training and test datasets from CPS datasets

1 $X_{train}, \mathbf{y}_{train}, X_{test}, \mathbf{y}_{test} \leftarrow TrainTestSplit(X, y)$

// Train the DL model employed for CPS data, h, with $(X_{train}, \mathbf{y}_{train})$

2 $f_p \leftarrow train(X_{train}, \mathbf{y}_{train})$

// Create an empty uncertainty dataset

3 $\mathcal{U} \leftarrow \emptyset$

// K ∈ {softmax variance, entropy}

4 **foreach** *uncertainty metric* $k \in K$ **do**

    // Calculate uncertainty metric k values for X_{test}

5     $\mathcal{U}[:, k] \leftarrow f_p.uncertainty(X_{test})$

// Obtain label predictions of h, i.e., $\hat{\mathbf{y}}$, for the test dataset. $(\hat{\mathbf{y}} == \mathbf{y}_{test})$ will be used as the label for uncertainty dataset $\mathcal{U}$.

6 $\hat{\mathbf{y}} \leftarrow f_p.predict(X_{test})$

// Build a prediction validator model, $h_{validator}$, with uncertainty dataset $\mathcal{U}$ and its label vector $(\hat{\mathbf{y}} == \mathbf{y}_{test})$

7 $h_{validator} \leftarrow SVM(\mathcal{U}, (\hat{\mathbf{y}} == \mathbf{y}_{test}))$

8 return $h_{validator}$

We designed our prediction validator for classification tasks. To extend it for regression tasks (i.e., continuous outputs), we need to adjust its implementation. For example, one can use the

standard error of the regression to create a training dataset, which describes the average distance that predicted values fall from the ground-truth regression line. This metric can label wrong predictions for the training dataset for the prediction validator model. More specifically, if a continuous output is higher than the standard error, we can label it as *wrong* prediction. We will implement such a prediction validator in the future and evaluate its effectiveness with relevant case studies.

5.2 Dropout Ratio Selection

Considering that a comprehensive approach would increase uncertainty values of incorrectly labeled input instances (i.e., FN and FP) while reducing uncertainty values of correctly labeled input instances (i.e., TP and TN), the model's prediction time dropout ratio, p, should be tuned for the base model to produce lower uncertainty values for true predictions and higher uncertainty values for false predictions. At the same time, this approach should maintain the prediction performance of the prediction model. Another parameter for estimating the best dropout ratio p value is model prediction performance. The F_1 value of the model is maximized to solve this optimization problem. The optimization problem's output is the optimal solution for the given model parameters. The equation is as follows:

$$\arg\min_{p} \Bigg(\frac{1}{m} \sum_{\mathbf{x}_i \in \mathcal{D}} \mathbb{I}\left(y_i = f_p(\mathbf{x}_i)\right) \mathcal{U}(f_p(\mathbf{x}_i)) + \frac{1}{n} \sum_{\mathbf{x}_i \in \mathcal{D}} \mathbb{I}\left(y_i \neq f_p(\mathbf{x}_i)\right) \frac{1}{\mathcal{U}(f_p(\mathbf{x}_i))} + \frac{1}{F_1^{(f_p)}} \Bigg), \tag{4}$$

where m is the number of correctly labeled examples, n is the number of incorrectly labeled examples in the unseen input space, $\mathcal{U}$ is the uncertainty quantification metric (i.e., entropy or prediction variance), and $F_1^{(f_p)}$ is the F_1 performance metric value of the model f. We used the greedy approach for the minimization of Equation (4) to estimate the best $p \in [0, 1]$ value for the optimal model classifier. The optimization problem is a simple minimization problem; there is only one variable, p, that justifies the sufficiency of using a greedy algorithm.

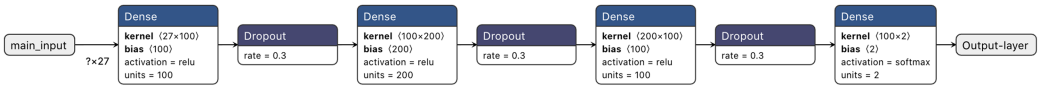
6 EVALUATION

In Section 6.1, we present the experiment design, followed by the experiment execution in Section 6.2 and results in Section 6.3. In Section 6.4, we present the threats to validity of our experiment.

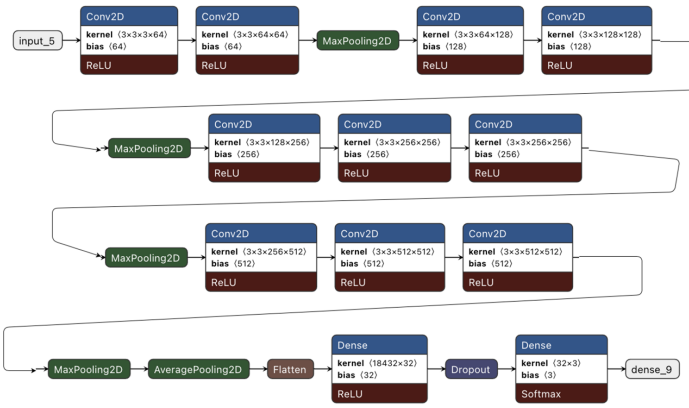
6.1 Experiment Design

We designed a series of experiments by uniformly varying 50 different p ratios from 0.01 to 0.90. Each experiment was executed 20 times, the results of which, corresponding to each p ratio, are averaged to smooth the plotting. We selected the best hyper-parameters for the DL models for the datasets we used in the experiments using a simple grid search, and it turned out that the best hyper-parameters are the Rmsprop optimizations with a learning rate of 0.01. Figure 4 shows the DL models for each dataset. The code is available from GitHub.[2]

[2]https://github.com/Simula-COMPLEX/nirvana.



(a) The activation function is relu for SWaT, sigmoid for Robot, and tanh for Video conferencing.



(b) The CNN based DL model for KITTI.

Fig. 4. Prediction-time-activated dropout-based neural network models for each dataset.

In the rest of the section, we first present the datasets we employed for the experiments in Section 6.1.1, followed by **research questions (RQs)** (Section 6.1.2) and evaluation metrics (Section 6.1.3).

6.1.1 Datasets. In this work, we used three publicly available and one industrial CPS datasets: SWaT [19], Robot [13], KITTI [38], and Video conferencing [44] to demonstrate the efficiency of NIRVANA.

The SWaT testbed implements the process for secure water treatment. The testbed has four security attack scenarios: *single-stage single-point, single-stage multi-point, multi-stage single-point, multi-stage multi-point*. The testbed consists of 51 sensors and actuators and their values and states are logged every second. The labeled data in this testbed consists of whether an attack occurred or not. To obtain such labels, various attacks according to the four scenarios were introduced to the testbed. The data we used is for over 11 days and it consists of both normal and attack scenarios.

The Robot dataset is the same as our running example presented in Section 3.2; however, the running example was simplified so that it can be used to explain things easily.

In the Video conferencing dataset, the data was collected about whether two Video conferencing systems can establish calls with each other under different configurations of the systems. The two Video conferencing systems are industrial CPSs that are available to us through our previous research collaborations.[3]

The KITTI dataset contains a series of computer vision tasks built using an autonomous driving platform. We used the KITTI road-type classification task [15]. The original dataset was designed for object detection and segmentation. However, the dataset has six different categories: *City*, *Residential*, *Road*, *Campus*, *Person*, and *Calibration*.[4] In our experiments, we classify road types of the KITTI dataset's images and consequently selected these three relevant categories: *Residential*, *City*,

[3]Note that due to the industrial nature of the data, we only provided password-protected access to the data only for the review. If the article is accepted, the data will not be made publicly available.
[4]http://www.cvlibs.net/datasets/kitti/raw_data.php.

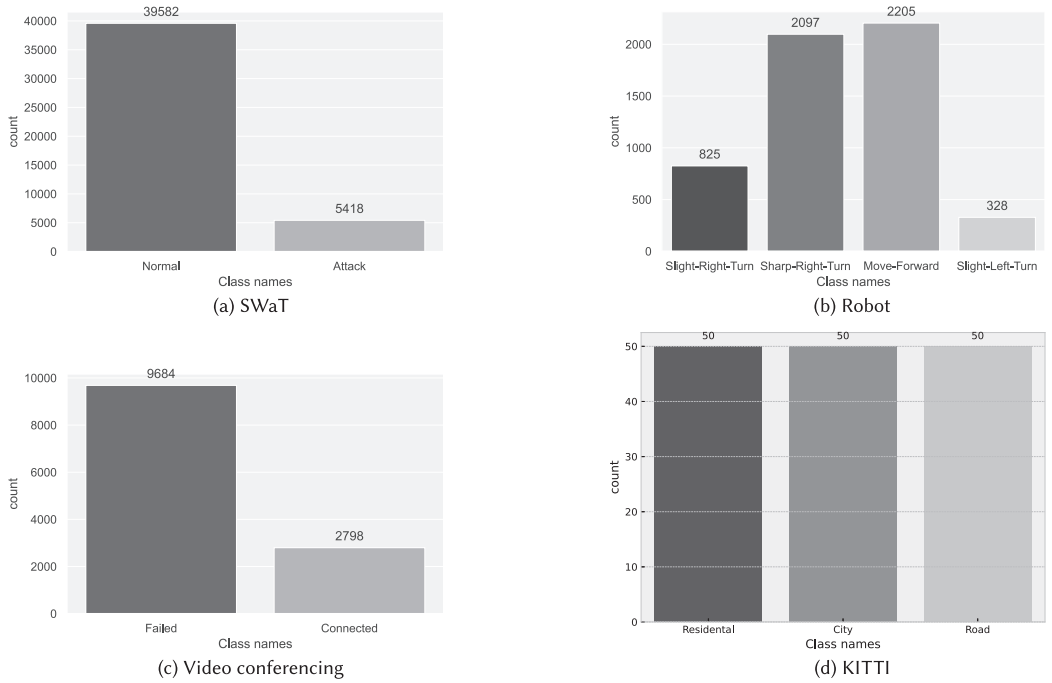


Fig. 5. Class label distributions of the datasets.

and *Road*. The other three categories are not relevant since their images contain either no roads or only a few walking paths. Figure 5 shows the class label distribution of the classes. We selected 50 images for each class to build the DL road classification model. The training time is quite high because of the images. Therefore, we could only select 150 images due to the practical constraints on the computational infrastructure we have, though the original KITTI dataset contains a fairly large number of images.

For the SWaT, Robot, and Video conferencing datasets, we developed feed-forward-based DL models for prediction. Since the KITTI dataset contains only autonomous car camera views (i.e., images), we developed a CNN-based DL model for KITTI to make predictions. In traditional image classification and object detection tasks, CNN-based DL models are most commonly used to solve problems that cannot be solved in feed-forward neural networks [27, 48].

Highly correlated features in a dataset are computationally expensive for the model building phase. To simplify it, we removed the highly correlated features from the SWaT, Robot, and Video conferencing datasets and trained each model using only low-correlated features to build more reliable and simple models. The resulting models are used in the evaluation phase to determine the prediction performance.

6.1.2 Research Questions. NIRVANA aims to explore uncertainty quantification, explore the relationship between prediction uncertainty and prediction accuracy, and create a prediction validator model based on this relationship. We formed the following RQs and designed the experiments to answer them: [**RQ0**] Is there any correlation between uncertainty and prediction performance? Studying this RQ is necessary as a positive answer to it motivates us to further investigate whether uncertainty values can help to improve the prediction performance of the DL models (RQ4). We numbered this research question as RQ0 since it is a simple question and the

existing literature has already answered this question to some extent [34, 54]. [**RQ1**] How can a model's decision making be characterized with uncertainty quantification? RQ1 helps us assess the quality of a DL model's decision making with two uncertainty quantification methods: entropy based and softmax prediction variance based. [**RQ2**] How can a model's false labeling be predicted by another model benefiting from uncertainty values (obtained from RQ1)? RQ2 explores the effectiveness of building a new model to extract decision boundaries (i.e., label clusters) using an RBF kernel-based SVM for validation data (i.e., prediction validator model). [**RQ3**] What is the best dropout ratio for uncertainty quantification for each dataset? RQ3 studies how to find the best dropout ratios to distinguish highly uncertain and wrong predictions from slightly uncertain and correct predictions. [**RQ4**] Can we improve the prediction performance of the DL models by using highly uncertain instances (obtained from RQ1) in the retraining of the models? RQ4 aims to demonstrate how to benefit from uncertainty values to build more reliable DL models.

So, overall, RQ0 was designed to confirm the negative correlation between uncertainty and prediction performance. RQ1 and RQ3 are about producing uncertainty values. RQ2 and RQ4 are for testing the effectiveness of using another model for predicting false labeling of DL models and improving their prediction performance, respectively, by both benefiting from uncertainty values obtained from RQ1 and RQ3.

6.1.3 Evaluation Metrics. As the datasets are not balanced and data is not uniform, we employ standard metrics in information retrieval to assess the models: the overall prediction accuracy, average recall, average precision [51], and F_1 [37]. We calculate precision and recall values for each class and then find their mean value.

6.2 Experiment Execution

All the experiments were performed using Python scripts and ML libraries: Keras, Tensorflow, and Scikit-learn, on the following machine: 2.8 GHz Quad-Core Intel Core i7 with 16GB of RAM.

For each dataset, two models, with and without MC dropout, were trained to obtain prediction results. In the first model, the dropout is applied at the prediction time. The hyper-parameters such as the number of hidden layers and the number of neurons in the hidden layers, the activation function, the loss function, and the optimization method are the same for both models.

Each dataset has been divided into two parts: 33% test sets and 67% training sets. In the test sets, we randomly chose a subset of the datasets and use them to evaluate the prediction performance of the models. The number of epochs was set to 500, and the epoch size was fixed for all the models. The prediction time dropout values were set to 0.01 and increased to 0.90 with an increment of 0.02.

We used dropout layers at the prediction time only. If we traditionally used the dropout method in training time, the DL models' neuron weights for each experiment (i.e., for different dropout values) could be different from each other's model. Here, we aim to carry out uncertainty measurements with varying dropout values of the DL model, whose neuron weights are very close to each other. If we use training time dropouts, then our DL models' weights can be very different in each experiment.

MC dropout models were used for each test input with $T = 100$ times for prediction of the test dataset. The predicted labels for the test set were calculated by averaging the output of the last layer's softmax function over T iterations. The prediction uncertainty for each input in the test sets was quantified using Shannon entropy and prediction variance, respectively.

6.3 Results

6.3.1 Results for RQ0. To answer this RQ, first, we obtain values of the prediction performance metrics (i.e., accuracy, precision, recall, and F_1) of the base NN models and the prediction-time-activated dropout-based NN models, as shown in Figure 2. Table 5 presents the prediction

Table 5. Prediction Performance Metrics of the Models

Dataset	Model	Acc.	Prec.	Recall	F_1
SWaT	Prediction-time-activated dropout-based NN	0.97	0.99	0.78	0.87
	Baseline model	0.97	0.99	0.79	0.88
Robot	Prediction-time-activated dropout-based NN	0.98	0.98	0.98	0.98
	Baseline model	0.97	0.97	0.97	0.97
Video conferencing	Prediction-time-activated dropout-based NN	0.81	0.80	0.81	0.80
	Baseline model	0.81	0.80	0.81	0.80
KITTI	Prediction-time-activated dropout-based NN	0.94	0.93	0.93	0.93
	Baseline model	0.94	0.93	0.93	0.93

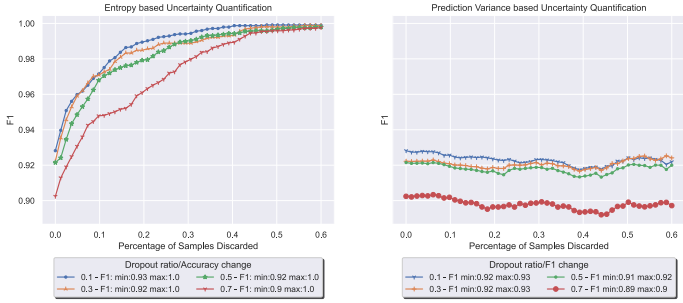
performance for each dataset of the two types of models. As shown in the table, the accuracy of the prediction-time-activated dropout-based NN model is roughly equal to that of the baseline model's prediction performance.

Second, all the samples were ranked in descending order according to their uncertainty values to find highly uncertain instances. Third, we discarded the topmost uncertain examples ranging from 0.1% to 60% in the datasets to demonstrate the negative effect of model uncertainty on prediction performance by removing the most uncertain samples from the datasets. Figure 6 shows changes in the prediction performances (measured with F_1) of the models with different dropout rates as a result of removing the most uncertain samples from each dataset. The left side of the figure shows the entropy-based uncertainty quantification results, whereas the right side of the figure shows the variance-based uncertainty quantification results. From the figure, we can observe that the prediction performance of the models increases along with discarding highly uncertain samples based on entropy-based uncertainty quantification. When looking at the results for variance-based uncertainty quantification, the trend is not as significant as for the entropy-based uncertainty quantification. In summary, the correlation between the entropy-based uncertainty quantification and the prediction performance is highly negatively correlated for most of the dropout rates and datasets, whereas no correlation between the variance-based uncertainty quantification and the prediction performance can be observed.

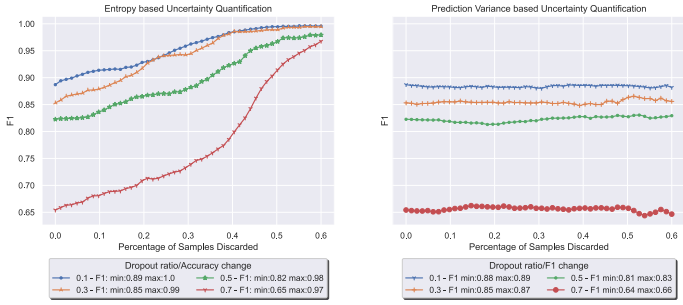
Another way to show DL model performance is ROC plots. In our case, the dropout ratio is varying. Thus, we need to plot 50 different ROCs for each dataset. Consequently, this approach is not suitable to show the relationship between uncertainty and prediction performance.

Concluding Remarks for RQ0: The prediction performance of models increases along with discarding highly uncertain samples based on entropy-based uncertainty quantification, which is, however, not always the case for variance-based uncertainty quantification.

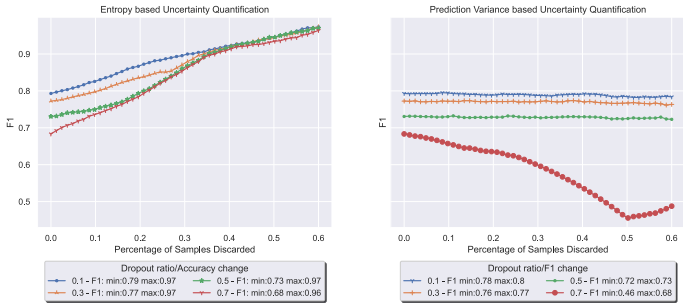
6.3.2 Results for RQ1. We used entropy and prediction variance values of the DL models' softmax probability outputs to compare each uncertainty quantification method to determine the most and least uncertain predictions on each dataset. Figures 7 to 10 show the two most uncertain test instances with entropy-based and softmax prediction probability variance-based uncertainty quantification for the four datasets. Each subfigure shows an input example with prediction probability



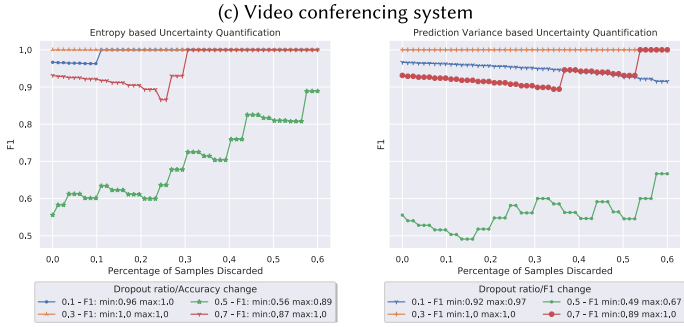
(a) SWaT



(b) Robot



(c) Video conferencing system

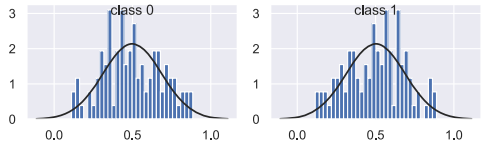


(d) KITTI

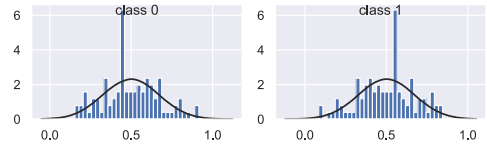
Fig. 6. Accuracy change with the most uncertain input instances removed from the test dataset.

(T times) histogram. The x-axis shows the softmax probability value of the class, and the y-axis shows its frequency.

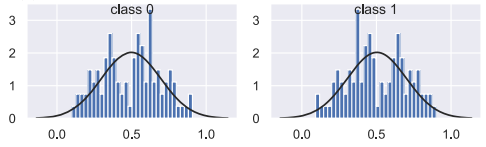
From these figures, one can observe that softmax probability output distributions for each class show very similar patterns in the results. When looking at Figure 9(c), the Video conferencing



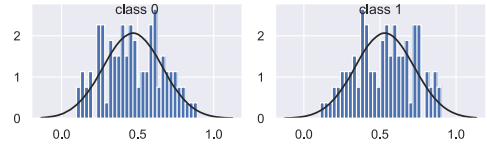
(a) **True**: 1, **Pred**: 1, **Class 0 prob**: 49.91% , **Class 1 prob:** 50.09% , **Ent**: 1.00



(b) **True**: 1, **Pred**: 1, **Class 0 prob**: 49.90% , **Class 1 prob**: 50.10% , **Ent**: 1.00

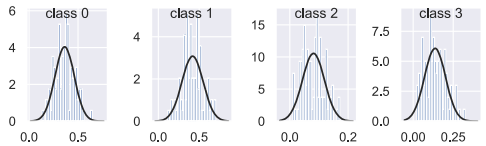


(c) **True**: 1, **Pred**: 1, **Class 0 prob**: 49.71%, **Class 1 prob**: 50.29%, **Pred.Var**: 0.20

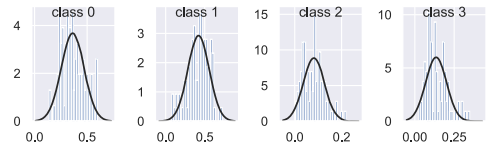


(d) **True**: 1, **Pred**: 1 ,**Class 0 prob**: 46.89%, **Class 1 prob**: 53.11%, **Pred.Var**: 0.19

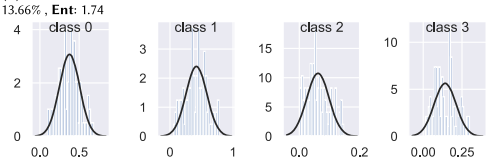
Fig. 7. Most uncertain test instances for the **SWaT** dataset. Figures (a) and (b) show results of the entropy-based uncertainty quantification, and Figures (c) and (d) show results of the softmax prediction probability variance-based uncertainty quantification. Class label 0 is `normal` and label 1 is `attack`.



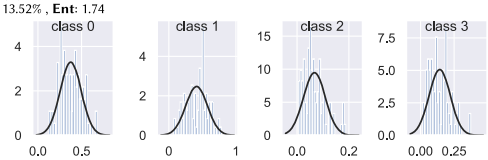
(a) **True**: 2 , **Pred**: 1 , **Class 0**: 36.33% , **Class 1**: 41.89% , **Class 2**: 8.11% , **Class 3**: 13.66% , **Ent**: 1.74



(b) **True**: 2 , **Pred**: 1 , **Class 0**: 36.22% , **Class 1**: 42.10% , **Class 2**: 8.16% , **Class 3**: 13.52% , **Ent**: 1.74



(c) **True**: 1 , **Pred**: 1 , **Class 0**: 37.72% , **Class 1**: 41.95% , **Class 2**: 6.21% , **Class 3**: 14.12% , **Pred.Var**: 0.10



(d) **True**: 1 , **Pred**: 1 , **Class 0**: 37.61% , **Class 1**: 41.40% , **Class 2**: 6.71% , **Class 3**: 14.28% , **Pred.Var**: 0.10

Fig. 8. Most uncertain test instances for the **Robot** dataset. Figures (a) and (b) show results of the entropy-based uncertainty quantification, and Figures (c) and (d) show results of the softmax prediction probability variance-based uncertainty quantification. Class label 0 is `Slight-Right-Turn`, label 1 is `Sharp-Right-Turn`, label 2 is `Move-Forward`, and label 3 is `Slight-Left-Turn`.

system model's prediction is 0, the true label is also 0 for the input example, and the uncertainty is very high (with the prediction variance being 0.15). Also, the class assignment probabilities for this input example are almost equal for each class label: 50.00% and 50.00%, indicating highly unreliable prediction. When looking at Figure 9(a), the Video conferencing system model's prediction is wrong, exhibiting the average prediction probability for 0-class being 49.94 %, and 1-class being 50.01% with an uncertainty value being 1 when measured with entropy.

In summary, the entropy and prediction softmax probability variance-based uncertainty quantification for the three datasets (i.e., SWaT, Robot, Video conferencing) show very similar patterns in the results in terms of uncertainty quantification and predictions' distributions, as Figures 7 to 9 show that the DL models' prediction conforms to the normal distribution. The softmax distribution of the predictions with high uncertainty generally shows a distribution with a midpoint of 0.5. Moreover, the average softmax prediction values of the class labels are also very close to each other. The only exception is for the KITTI dataset, as shown in Figure 10. We can see from the figure that, in general, the KITTI DL model's predictions have low uncertainty, which

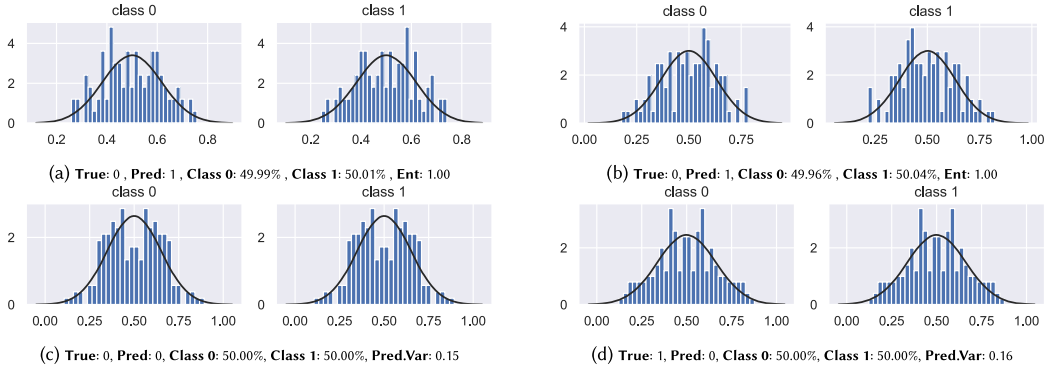


Fig. 9. Most uncertain test instances for the **Video conferencing system** dataset. Figures (a) and (b) show results of the entropy-based uncertainty quantification results, and Figures (c) and (d) show results of the softmax prediction probability variance-based uncertainty quantification. Class label 0 is `Failed` and label 1 is `Connected`.

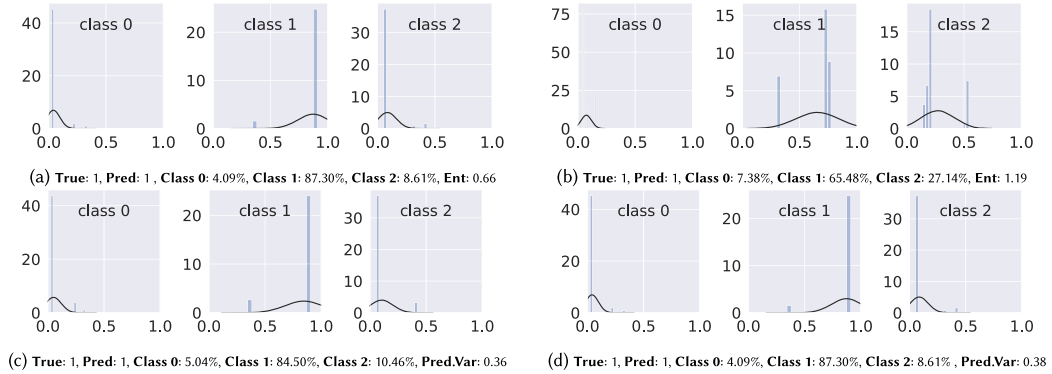


Fig. 10. Most uncertain test instances for the **KITTI** dataset. Figures (a) and (b) show results of the entropy-based uncertainty quantification, and Figures (c) and (d) show results of the softmax prediction probability variance-based uncertainty quantification. Class label 0 is `Residental`, label 1 is `City`, and label 2 is Road.

is also supported from data reported in Table 5, where the precision, recall, and F1 prediction performance values of the KITTI DL model are higher than the other models.

Concluding Remarks for RQ1: The entropy- and softmax probability variance-based methods perform similarly in terms of uncertainty quantification and predictions' distributions, suggesting that highly uncertain instances' predictions are likely wrong.

6.3.3 Results for RQ2. We can predict if the labeling is FP or FN with another prediction model using entropy and variance uncertainty values. For this purpose, for each dataset, we created a dedicated SVM-based prediction validator model with the following parameter configurations: `RBF kernel`, `Gamma` value being 1,000, `Regularization parameter (C)` being 1,000. The parameter values were chosen to maximize the prediction validator accuracy based on grid search. The input data of these prediction models are the entropy and variance values of each instance in the training datasets; the labels are correct prediction (0) or wrong prediction (1). These models learn decision boundaries from the training datasets followed by predicting wrong predictions based on regions.

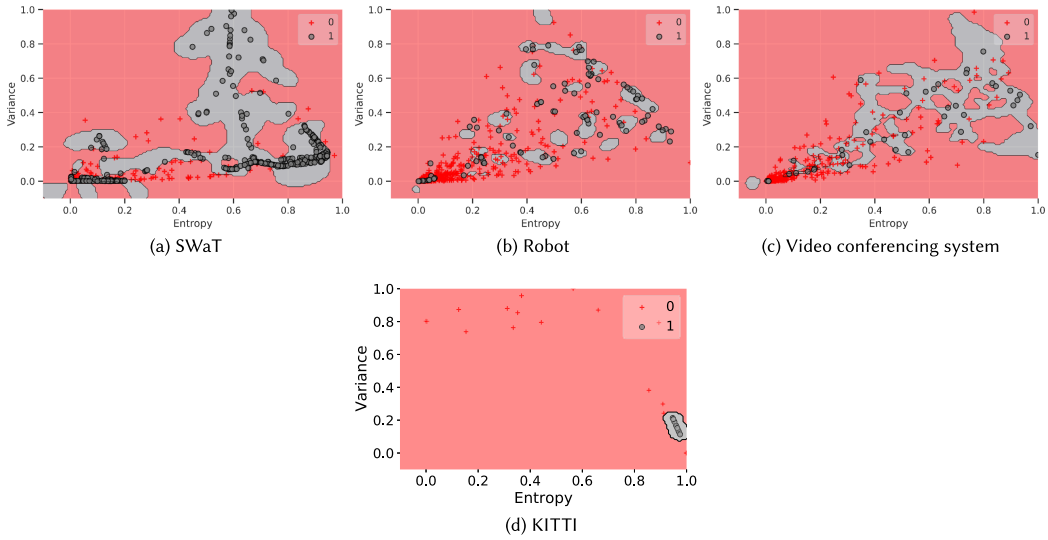


Fig. 11. Decision boundaries of the prediction validator models for each dataset.

Table 6. SVM Model Prediction Performance

Data	RBF				Linear				Polynomial			
	Prec.	**Rec.**	**F_1**	**Acc.**	**Prec.**	**Recall**	**F_1**	**Acc.**	**Prec.**	**Rec.**	**F_1**	**Acc.**
Robot	0.97	0.95	0.96	0.97	0.81	0.81	0.81	0.81	0.77	0.77	0.77	0.77
SWaT	0.94	0.90	0.92	0.95	0.84	0.84	0.83	0.84	0.79	0.70	0.71	0.78
Video	0.96	0.93	0.94	0.94	0.77	0.77	0.77	0.77	0.74	0.75	0.73	0.75
KITTI	0.97	0.97	0.97	0.97	0.95	0.95	0.95	0.95	0.88	0.85	0.85	0.85

One important feature of a prediction validator model is that it can form decision boundaries that distinguish among classes in a dataset in the form of polynomial/linear functions and clusters [25]. In particular, a class's boundary is defined as a set of clusters with all the instances having the same label. Figure 11 shows decision boundaries for positive and negative classes separated as clusters. The grey regions cover positive labeled instances (FN and FP predictions for the base NN model), while the red regions show negative labeled instances (TP and TN predictions for the base NN model). In other words, if the entropy and variance uncertainty values of a test instance are in the grey region of the decision boundary, the prediction validator model will classify this instance as a wrong prediction.

As shown in Figure 11, there are six clusters (separated grey regions) for SWaT, around 20 clusters for Robot, 2 clusters for Video conferencing, and 1 cluster for KITTI. The prediction validator model for KITTI results in only one cluster, implying that the prediction validator models with the RBF kernel can easily distinguish positive and negative instances. Moreover, Table 6 shows the prediction performance results with three different kernel functions: *RBF*, *Linear*, and *Polynomial*. From the table, we observe that the RBF-based SVM model for each dataset achieved the highest classification results. More details about the comparisons will be discussed in Section 7.

Table 7 presents results of the comparison of the threshold-based wrong label prediction and SVM-based wrong label prediction. We aim to compare the prediction performance of the prediction validator and threshold-based wrong label prediction method. To make a fair comparison, we

Table 7. Comparison against the Current Standard in Terms of Entropy-based Uncertainty Quantification

Data	**Threshold (Entropy)** Based			**SVM Based**	
	Threshold	FPR	TPR	FPR	TPR
Robot	0.763	0.094634	0.752941	0.094718	0.88591
SWat	0.373	0.139068	0.566565	0.138698	0.613824
Video	0.587	0.105154	0.844735	0.105459	0.854712
KITTI	0.910	0.082192	0.571429	0.082278	0.95302

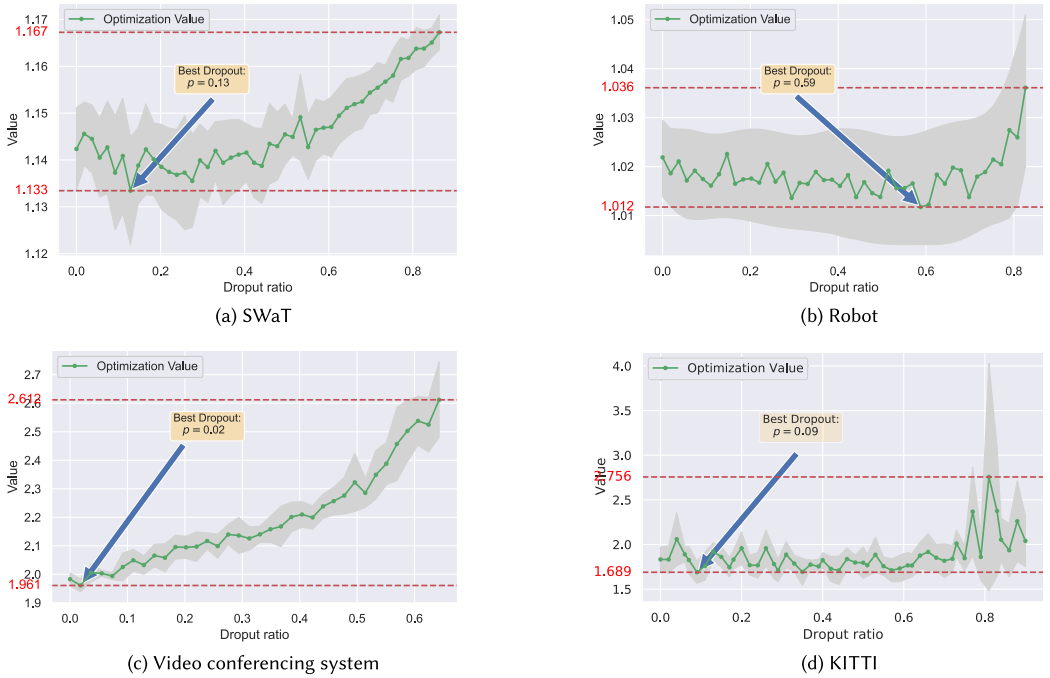
used the **false-positive rate (FPR)** metric to find the matching threshold value. In this way, we developed two different models that have the same FPR value. In the threshold-based wrong label prediction approach, if a prediction's entropy-based uncertainty value is higher than a predefined value (i.e., threshold value), this prediction is accepted as wrong. Accordingly, we used the 500 different threshold values between the minimum and maximum uncertainty values. In the second step, we calculated the corresponding FPR and TPR values for each threshold value. Finally, the threshold with the closest FPR rate with the SVM FPR rate is selected. The matching threshold values are 0.763 for Robot, 0.373 for SWat, 0.587 for Video conferencing, and 0.910 for KITTI datasets. Though the FPR values are almost the same, the TPR values are higher with our SVM-based method. Therefore, NIRVANA gives better prediction performance than threshold-based wrong label predictions.

The training dataset used to train a DL model may not contain enough examples and may not cover all situations for a given task. In this case, the DL model's prediction uncertainty value would be high. For this reason, unfamiliar input to the training set will be predicted as wrong by the prediction validator as it has high uncertainty in its prediction. Moreover, the DL model's prediction will probably be incorrect. As can be seen in Table 6, the performance of the prediction validator model to detect incorrect predictions is quite high, with, for instance, F_1 scores between 0.92 and 0.97. As a result, the prediction validator model successfully extracts the relationship between uncertainty measurements and incorrect predictions.

Concluding Remarks for RQ2: In most cases, the prediction validator model can extract relationships between uncertainties and incorrect predictions with high accuracy (more than 0.94), recall (at least 0.90), precision (more than 0.97), and F_1 score (at least 0.92), thereby providing a valuable tool for uncertainty analyses related to incorrect predictions.

6.3.4 Results for RQ3. For RQ3, we aim to find the best dropout ratios, p, using Equation (4) for each dataset. We conducted all experiments using the test datasets to find the best p values. Figure 12 shows the output (y-axis) of the equation with respect to p. According to the figures, the best p value is 0.13 for SWaT, 0.59 for Robot, 0.02 for Video conferencing, and 0.09 for KITTI datasets. We observed that the best p value selected for Robot is higher than for the other datasets. This is because, from the figure, we can see that the standard deviation of the outputs of Equation (4) with varying p values (i.e., the dark grey area) for Robot is large. One can also observe that the outputs increase until the p value reaches 0.59. Therefore, we think that choosing a lower value for p as the best dropout ratio will not make much difference in uncertainty quantification.

Figure 13 shows the CPS data models' entropy-based uncertainty quantification with the best p values for each dataset. The figure has the same shape for all four datasets. In each figure, the x-axis shows the model uncertainty value and the y-axis shows the density of data points. The green lines are the TP and TN values of each model. For instance, Figure 13(a) shows the uncertainty histogram



(a) SWaT (b) Robot

(c) Video conferencing system (d) KITTI

Fig. 12. Best dropout ratios according to Equation (4).

of the SWaT dataset. Its sub-figure on the left shows the uncertainty of positive examples (true *Attack* labels) of predictions with TP and FP labels, respectively. The TP and TN labeled instances' uncertainty histogram values are clustered in the lower values (the left side of the figures), which means low uncertainty on correctly labeled samples.

On the other hand, the FP and FN labeled instances' uncertainty histogram values are more scattered in the upper values (the right side of the figures), indicating high uncertainty in incorrectly labeled samples. This pattern can be observed for all four datasets. The only exception is on the TN predictions for the Video conferencing dataset. As shown in Figure 13(c), the uncertainty histogram values for the TN predictions (in the *Connected* class) with FN predictions are clustered in higher uncertainty values (the right side of the figure), indicating high uncertainty both in incorrect and in correctly labeled samples. The main reason is that the Video conferencing dataset's instances with the negative labels have a similar pattern as instances with the positive labels. Wrong predictions for the KITTI dataset are only in the *Road* class. There are no wrong predictions in the other two classes. The Road class's histogram plot shows a similar pattern as the other datasets' prediction histogram plots; the uncertainty values of correct predictions are on the left side, while the uncertainty values of wrong predictions are on the right side of the figure.

In conclusion, the uncertainty values of the correct and incorrect predictions are clustered in different regions in the figures. The correct predictions' uncertainty values are on the left side (i.e., low uncertainty), and the wrong predictions are on the right side (high uncertainty).

Concluding Remarks for RQ3: The selected best dropout ratio for each dataset (0.13 for SWaT, 0.59 for Robot, 0.02 for Video conferencing, and 0.09 for KITTI) successfully increases the wrongly labeled instances' uncertainty values while reducing the correctly labeled instances' uncertainty values.

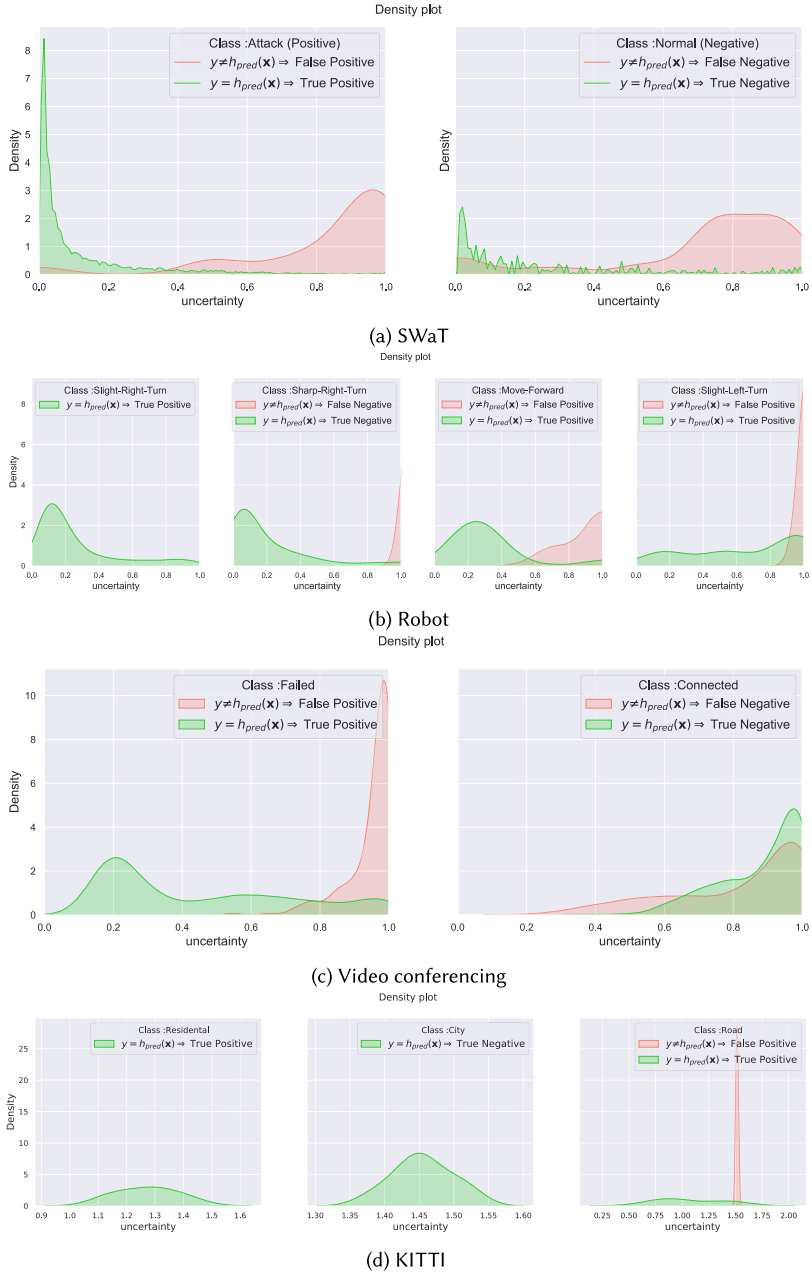
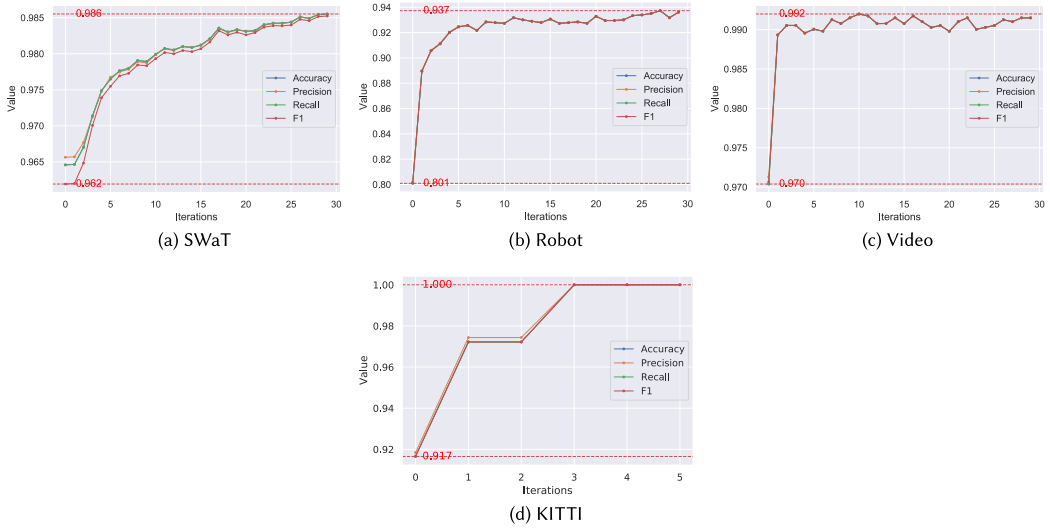


Fig. 13. Prediction-time-activated dropout-based DL model uncertainty for all datasets.

6.3.5 Results for RQ4. We have shown in RQ0 that samples with high uncertainty have a high rate of wrong predictions. Highly uncertain samples can be used for retraining to improve the DL models' prediction performance. To achieve this, we iteratively retrained each DL model. At each retraining iteration, predictions were made using the test datasets, and we further ranked these predictions in descending order according to their uncertainty values. We used the most



(a) SWaT (b) Robot (c) Video

(d) KITTI

Fig. 14. Performance improvements of the DL models when retrained with most uncertain input instances. F_1 scores before and after retraining each DL model are highlighted in each sub-figure.

Table 8. Accuracy Improvement of the DL Models When Retrained with Most Uncertain Input Instances

	Accuracy		**Precision**		**Recall**		**F1**	
	Before	After	Before	After	Before	After	Before	After
SWaT	0.96457912	0.98552189	0.96563238	0.98546561	0.96457912	0.98552189	0.96191914	0.985234
Robot	0.80122154	0.93725708	0.80191343	0.93729346	0.80122154	0.93725708	0.80088805	0.93724854
Video	0.97038835	0.99199029	0.97128911	0.99198228	0.97038835	0.99199029	0.9706586	0.99198553
KITTI	0.91666667	1.0	0.91841492	1.0	0.91666667	1.0	0.91652174	1.0

uncertain instances in the next iteration of retraining, and consequently improved the prediction performance of the DL models.

To converge each iteration, the number of new rows in retraining was selected using a decaying approach, as shown below:

$$num\_of\_rows = \frac{\sum_{\mathbf{x}_i \in \mathcal{D}} \mathbb{I}\left(y_i \neq f_p(\mathbf{x}_i)\right)}{1 + \alpha \cdot iter\_no}, \tag{5}$$

where y_i is the actual output, $f_p(\mathbf{x}_i)$ is the predicted output, and α is the tuning parameter. The α value is 1.0 for the SWaT, Robot, and Video conferencing datasets, and 0.2 for the KITTI dataset. The KITTI dataset's number of instances is lower than the other datasets. Thus, we aimed for the iteration steps to take longer by choosing a lower α value. In this way, we were able to have more $num\_of\_instances$ values in each iteration step. Figure 14 shows the prediction performance improvement for each dataset. The SWat, Robot, and Video conferencing DL models' iteration sizes are 30, while the KITTI DL model's is 5 because, as shown in the figure, the KITTI DL model converges to 1.0 within five iterations.

Table 8 summarizes the prediction performance improvement of each DL model. According to the table, all the DL models' prediction performance increases after being retrained with highly uncertain training instances provided with our approach. For instance, the prediction performance improvements of the DL models in terms of accuracy are 2.17% for SWaT, 16.98% for Robot, 2.23% for Video conferencing, and 9.09% for KITTI.

Concluding Remarks for RQ4: Retraining DL models with highly uncertain training instances increases their prediction performance by 2% to 17% on all four metrics. These results suggest that the prediction performance of a DL model can be improved by retraining the model with uncertain instances.

6.4 Threats to Validity

A key *external validity* threat is related to the generalization of results [43]. In our experiments, we used only four CPS datasets, and we definitely need more case studies to generalize the results. Moreover, these datasets reflect different types of CPSs, including robotics, automation systems, and communication systems.

Our key *construct validity* threat is related to the selection of uncertainty quantification metrics that do not precisely quantify uncertainty. Nevertheless, note that these quantification metrics are from the literature [43] and have been applied to quantify uncertainty. In the future, we will conduct dedicated empirical studies to systematically investigate more uncertainty quantification metrics. Another threat is to use the SVM algorithm for the prediction validator model. One reason was that the SVM models are capable of finding nonlinear decision boundaries in input space [24]. Moreover, the SVM algorithm is capable of separating class instances using a hyperplane with kernel trick (i.e., imitating higher-dimensional input space). Nonetheless, in the future, we will conduct experiments to build a prediction validator to investigate more classification algorithms. The last threat is related to activating the dropout layers in the prediction time only. Because of this, our approach is not fully compatible with Gal's approach in terms of quantifying uncertainty in DL models. Also, we developed our implementation without using any uncertainty quantification library. Being now aware that the uncertainty-wizard library [57] makes uncertainty quantification compatible with BNN models, we will carry out experiments by using BNN models and the uncertainty-wizard library in the future.

Our main *conclusion validity* threat is due to finding the best dropout ratio that is responsible for increasing uncertainty for the wrong labeling instances and decreasing the uncertainty for the correctly labeled instances. To mitigate this threat, we repeated each experiment 20 times for each dataset to reduce the probability that the results were obtained by chance. In a standard neural network training, all weights are initialized uniformly at random. In the second stage, using optimization, these weights are updated to fit the classification problem. Since the training started with a probabilistic approach, there is a possibility of facing the local minimum problem in the optimization. In order to eliminate the local minimum problem, we repeated the training 20 times to find the p value that gives the best result. In each repetition, the weights were initialized uniformly at random but with different values, such that if the optimization function failed to find the global minimum, in the next experiment, it is likely to find it as the weights have been initialized with different values.

7 OVERALL DISCUSSION

In this section, we provide overall discussion from two perspectives. First, we present the discussion related to detecting wrong predictions in Section 7.1, followed by discussing applications of NIRVANA in the context of studied CPSs in Section 7.2.

7.1 Detection of Wrong Predictions

This study demonstrated that there is a strong relationship between DL models' prediction accuracy and predictive uncertainty with various CPS datasets. In existing uncertainty-based prediction

quality studies [17, 39], a model's prediction reliability for a given instance is generally decided using only a threshold value. If a prediction's uncertainty value is higher than a predefined threshold value, the system concludes that this prediction is likely wrong. Our study shows that this relationship is not simply a threshold or a linear correlation. We created uncertainty datasets for each DL model employed for CPS data using the models' prediction uncertainty over the entire input CPS datasets. Labels of such an uncertainty dataset are binary classes of correct predictions (TP and TN) and wrong predictions (FP and FN). In this dataset, we observed that correct and incorrect predictions are clustered into specific regions. For this reason, we built an SVM model with RBF kernel that creates region-based decision boundaries in the uncertainty datasets. Thus, we increased the performance of the detection of wrong predictions as shown in Table 6.

In this study, the prediction validator estimates wrong predictions of classification models using uncertainty metrics. Apart from classifications, a regression model's wrong predictions can be calculated in the same way by using error metrics such as mean absolute error, mean square error, and root mean square error. Therefore, the prediction validator approximation is suitable not only for classification but also for regression problems.

7.2 Applications in the Context of the Studied CPS Datasets

Below, we discuss such applications for each of the studied CPS datasets and the corresponding underlying CPS.

SWaT: The SWaT CPS case study is focused on the detection of security attacks. To this end, uncertainty datasets produced with NIRVANA can be used as training data or testing input to facilitate the development and testing of an uncertainty-aware DL-based attack detection system that can predict attacks in various uncertain situations such as due to sensor and actuator errors. The aim of such attacks is to deceive the system and lead it to anomalous behavior. When the decision of the DL model employed for CPS data is wrong, it makes the target system fail (e.g., wrong behavior of actuators). In the studies about "Adversarial machine learning" attacks such as FGSM [20], BIM [30], and DeepFool [40], the uncertainty of the DL models' decisions as a result of the attacks is high [12]. For this reason, the use of uncertainty datasets in CPSs' attack detection will increase the reliability of the system.

Video conferencing system: For a CPS to work correctly and efficiently, it is essential that its system parameters are properly configured during its operation. A video conferencing system running with incorrectly configured parameters is highly likely to fail. In this context, we foresee two possible applications. First, an uncertainty-aware DL-based component can be developed to predict call failures ahead of time due to uncertain configurations of the video conferencing system. Such a component, in addition, can produce quantified uncertainty associated with each prediction, which can be used by the system to make better decisions such as keeping the call connection but reducing the quality of videos of call participants automatically. Second, an uncertainty-aware method, by benefiting from uncertainty datasets, for testing videoconferencing systems under various configurations can be developed.

Robot: In the context of this system, we can possibly develop DL-based systems to predict the next move of the robot even in uncertain situations such as obstacles with quantified uncertainty for each move decision. If decisions made by an autonomous robot system involve a high level of uncertainty, such decisions are probably wrong. It is critical for autonomous robots to measure the quality of their decisions during navigation and be aware that their decisions may be wrong and might need to pursue human intervention. For example, we think it can be used for autonomous robots that can be used in extraterrestrial environments such as the Moon and Mars or deep ocean research.

KITTI: The KITTI case study is focused on the detection of road types from autonomous cars' cameras. We can possibly develop DL-based systems to predict road types in uncertain situations such as obstacles with quantified uncertainty for each steering wheel angle decision. Detecting the prediction error made by the DL model employed for CPS data and warning the CPS operator are of vital importance in many cases to avoid fatal crashes. Examples of autonomous cars making wrong decisions include detecting lanes incorrectly and outputting wrong steering angle and braking commands. Such wrong decisions often have a high degree of associated uncertainty, if it is measured. Figure 11(d) shows the prediction validator's decision boundaries based on entropy and variance uncertainty values. While the entropy value is high, the variance value is low in some instances. The opposite is also observed in some other instances, indicating that wrong predictions may not have high uncertainty as shown in Section 6.3.5. As we have shown in our experiments, it is necessary to use another prediction model to derive this uncertainty and prediction error relationship.

8 CONCLUSION AND FUTURE WORK

In this study, we have discussed five main issues related to uncertainty quantification, its correlation with prediction performance, and prediction validator, namely: (1) How can the model's decision making be characterized by uncertainty quantification? (2) Is uncertainty quantification correlated with the prediction performance of a model? (3) How to predict the model's wrong labeling using uncertainty values? (4) What is the strategy for selecting the best dropout ratio? (5) Can highly uncertain instances be used to improve prediction performance of DL models? We conducted experiments with four real-world CPS datasets to answer these questions. Our results confirm that highly uncertain predictions are likely wrong as we observed that uncertainty metrics are important indicators to show that a DL model makes a wrong decision. Our empirical results also show that the selected dropout ratio for each dataset successfully increases the wrongly labeled instances' uncertainty values while reducing the correctly labeled instances' uncertainty values. In addition, the most accurate classifiers could be obtained with low dropout ratios, suggesting that it is possible for the strategy to improve the classifier's performance.

All in all, we conclude that this experimental work has given us essential insights into the properties of epistemic uncertainty in DL for CPS data. We built a highly discriminating prediction validator model. We also suggest ways for the validation of our strategy for estimating the best dropout ratio. The current work highlights the need for systematic approaches to improve the predictability of the models for small samples, with a focus on CPS datasets. In this study, we tested feed-forward and CNN-based DL model architectures. We will also test more complex DL models to measure the efficiency of our method. In particular, our next step will be to investigate the generation of the highly uncertain samples with wrong predictions to improve the prediction performance. Our second step is to study the Deep Ensemble-based uncertainty quantification in DL models. We want to show the differences between MC dropout and Deep Ensemble for CPS datasets. Lastly, NIRVANA is currently applicable for classification tasks. There are also regression models that have continuous variables in CPSs. In future work, we want to apply the prediction validator method to continuous variables. Another future work is to use uncertainty-ranking-based test case generation and prioritization to test DL models. We believe that uncertainty-ranking-based test case generation would improve prediction performance of DL models. In addition to our current SVM-based prediction validator, in the future, we also aim to try to build the validator with other ML algorithms (e.g., DL, MLP, Random Forests) and conduct empirical studies to evaluate their performance. This approach can also be employed in other fields other than CPSs, which can be another future work.

REFERENCES

[1] Sara Abbaspour Asadollah, Rafia Inam, and Hans Hansson. 2015. A survey on testing for cyber physical system. In *Testing Software and Systems*, Khaled El-Fakih, Gerassimos Barlas, and Nina Yevtushenko (Eds.). Springer International Publishing, Cham, 194–207.

[2] Luciano Baresi, Liliana Pasquale, and Paola Spoletini. 2010. Fuzzy goals for requirements-driven adaptation. In *2010 18th IEEE International Requirements Engineering Conference*. IEEE, 125–134.

[3] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador Garcia, Sergio Gil-Lopez, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. 2020. Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion* 58 (2020), 82–115. https://doi.org/10.1016/j.inffus.2019.12.012

[4] David Berend, Xiaofei Xie, Lei Ma, Lingjun Zhou, Yang Liu, Chi Xu, and Jianjun Zhao. 2020. Cats are not fish: Deep learning testing calls for out-of-distribution awareness. In *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE'20)*. 1041–1052.

[5] Vanessa Buhrmester, David Münch, and Michael Arens. 2019. Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey. (2019). arXiv:cs.AI/1911.12116.

[6] Matteo Camilli, Carlo Bellettini, Angelo Gargantini, and Patrizia Scandurra. 2018. Online model-based testing under uncertainty. In *2018 IEEE 29th International Symposium on Software Reliability Engineering (ISSRE'18)*. IEEE, 36–46.

[7] Ferhat Ozgur Catak, Tao Yue, and Shaukat Ali. 2021. Prediction surface uncertainty quantification in object detection models for autonomous driving. In *2021 IEEE International Conference on Artificial Intelligence Testing (AITest'21)*. 93–100. https://doi.org/10.1109/AITEST52744.2021.00027

[8] Betty H. C. Cheng, Pete Sawyer, Nelly Bencomo, and Jon Whittle. 2009. A goal-based modeling approach to develop requirements of an adaptive system with environmental uncertainty. In *International Conference on Model Driven Engineering Languages and Systems*. Springer, 468–483.

[9] M. Combalia, F. Hueto, S. Puig, J. Malvehy, and V. Vilaplana. 2020. Uncertainty estimation in deep neural networks for dermoscopic image classification. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'20)*. IEEE, 3211–3220.

[10] Tommaso Dreossi, Daniel J. Fremont, Shromona Ghosh, Edward Kim, Hadi Ravanbakhsh, Marcell Vazquez-Chanlatte, and Sanjit A. Seshia. 2019. VerifAI: A toolkit for the formal design and analysis of artificial intelligence-based systems. In *31st International Conference on Computer Aided Verification (CAV'19)*. Springer International Publishing, 11.

[11] Omer Faruk Tuna, Ferhat Ozgur Catak, and M. Taner Eskil. 2022. Closeness and uncertainty aware adversarial examples detection in adversarial machine learning. *Computers and Electrical Engineering*. 101 (2022), 107986. https://www.sciencedirect.com/science/article/pii/S0045790622002555.

[12] Omer Faruk Tuna, Ferhat Ozgur Catak, and M. Taner Eskil. 2022. Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples. *Multimedia Tools Appl.* 81, 8 (2022), 11479–11500. https://doi.org/10.1007/s11042-022-12132-7

[13] A. L. Freire, G. A. Barreto, M. Veloso, and A. T. Varela. 2009. Short-term memory mechanisms in neural network learning of robot navigation tasks: A case study. In *2009 6th Latin American Robotics Symposium (LARS'09)*. IEEE, 1–6. https://doi.org/10.1109/LARS.2009.5418323

[14] Yarin Gal. 2016. Uncertainty in deep learning. *University of Cambridge* 1, 3 (2016), 174.

[15] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. 2013. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research* 32, 11 (2013), 1231–1237. https://doi.org/10.1177/0278364913491297.

[16] Eva Geisberger and Manfred Broy (Eds.). 2015. *Living in a Networked World: Integrated Research Agenda Cyber-Physical Systems (agendaCPS'15)*. http://www.acatech.de/de/publikationen/empfehlungen/acatech/detail/artikel/living-in-a-networked-world-integrated-research-agenda-cyber-physical-systems-agendacps.html.

[17] Biraja Ghoshal, Cecilia Lindskog, and Allan Tucker. 2020. Estimating uncertainty in deep learning for reporting confidence: An application on cell type prediction in testes based on proteomics. In *Advances in Intelligent Data Analysis XVIII*, Michael R. Berthold, Ad Feelders, and Georg Krempl (Eds.). Springer International Publishing, Cham, 223–234.

[18] Biraja Ghoshal and Allan Tucker. 2020. Estimating Uncertainty and Interpretability in Deep Learning for Coronavirus (COVID-19) Detection. (2020). arXiv:eess.IV/2003.10769.

[19] Jonathan Goh, Sridhar Adepu, Khurum Nazir Junejo, and Aditya Mathur. 2016. A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*. Springer, 88–99.

[20] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv e-prints* 1, 1, Article arXiv:1412.6572 (Dec. 2014), 11 pages. arXiv:stat.ML/1412.6572.

[21] Xiaozhe Gu and Arvind Easwaran. 2019. Towards safe machine learning for CPS: Infer uncertainty from training data. In *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS'19)*. Association for Computing Machinery, New York, NY, 249–258. https://doi.org/10.1145/3302509.3311038

[22] David Gunning and David Aha. 2019. DARPA's explainable artificial intelligence (XAI) program. *AI Magazine* 40, 2 (2019), 44–58.
[23] S. C. Gupta and V. K. Kapoor. 2020. *Fundamentals of Mathematical Statistics*. Sultan Chand & Sons.
[24] Jiawei Han, Micheline Kamber, and Jian Pei. 2012. 9 - Classification: Advanced methods. In *Data Mining* (3rd ed.), Jiawei Han, Micheline Kamber, and Jian Pei (Eds.). Morgan Kaufmann, Boston, 393–442. https://doi.org/10.1016/B978-0-12-381479-1.00009-5
[25] S. Han, Cao Qubo, and Han Meng. 2012. Parameter selection in SVM with RBF kernel function. In *World Automation Congress 2012*. IEEE, 1–4.
[26] Maximilian Henne, Adrian Schwaiger, Karsten Roscher, and Gereon Weiss. 2020. Benchmarking uncertainty estimation methods for deep learning with safety-related metrics. *Workshop on Artificial Intelligence Safety (SafeAI 2020)*. 83–90.
[27] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. 2016. Densely connected convolutional networks. *arXiv e-prints*, Article arXiv:1608.06993 (Aug. 2016), arXiv:1608.06993 pages. arXiv:cs.CV/1608.06993.
[28] Lazaros Iliadis, Stavros Tachos, Stavros Avramidis, and Shawn Mansfield. 2011. Support vector machines versus artificial neural networks for wood dielectric loss factor estimation. In *Engineering Applications of Neural Networks*, Lazaros Iliadis and Chrisina Jayne (Eds.). Springer, Berlin, 140–149.
[29] Alex Kendall and Yarin Gal. 2017. What uncertainties do we need in Bayesian deep learning for computer vision? In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, 5580–5590.
[30] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2016. Adversarial examples in the physical world. *arXiv e-prints*, Article arXiv:1607.02533 (July 2016), arXiv:1607.02533 pages. arXiv:cs.CV/1607.02533.
[31] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. 2017. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Curran Associates Inc., Red Hook, NY, 6405–6416.
[32] Imane Lamrani, Ayan Banerjee, and Sandeep K. S. Gupta. 2020. Toward operational safety verification via hybrid automata mining using I/O traces of AI-enabled CPS. In *Workshop on Artificial Intelligence Safety (SafeAI 2020)*. 186–194.
[33] Christian Leibig, Vaneeda Allken, Murat Seçkin Ayhan, Philipp Berens, and Siegfried Wahl. 2017. Leveraging uncertainty information from deep neural networks for disease detection. *Scientific Reports* 7, 1 (2017), 1–14.
[34] Antonio Loquercio, Mattia Segu, and Davide Scaramuzza. 2020. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters* 5, 2 (2020), 3153–3160. https://doi.org/10.1109/LRA.2020.2974682
[35] Tao Ma, Shaukat Ali, and Tao Yue. 2021. Testing self-healing cyber-physical systems under uncertainty with reinforcement learning: An empirical study. *Empirical Software Engineering* 26, 3 (2021), 1–54.
[36] Wei Ma, Mike Papadakis, Anestis Tsakmalis, Maxime Cordy, and Yves Le Traon. 2021. Test selection for deep learning systems. *ACM Trans. Softw. Eng. Methodol.* 30, 2, Article 13 (Jan. 2021), 22 pages. https://doi.org/10.1145/3417330
[37] John Makhoul, Francis Kubala, Richard Schwartz, and Ralph Weischedel. 1999. Performance measures for information extraction. In *In Proceedings of DARPA Broadcast News Workshop*. 249–252.
[38] Moritz Menze and Andreas Geiger. 2015. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR'15)*.
[39] Rhiannon Michelmore, Marta Kwiatkowska, and Yarin Gal. 2018. Evaluating uncertainty quantification in end-to-end autonomous driving control. *CoRR* abs/1811.06817 (2018). arXiv:1811.06817 http://arxiv.org/abs/1811.06817.
[40] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard. 2016. DeepFool: A simple and accurate method to fool deep neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*. 2574–2582. https://doi.org/10.1109/CVPR.2016.282
[41] Narendra Patwardhan and Zequn Wang. 2019. Reinforcement learning for robotics and control with active uncertainty reduction. *CoRR* abs/1905.06274 (2019). arXiv:1905.06274. http://arxiv.org/abs/1905.06274.
[42] Jana-Rebecca Rehse, Nijat Mehdiyev, and Peter Fettke. 2019. Towards explainable process predictions for industry 4.0 in the DFKI-smart-Lego-factory. *KI-Künstliche Intelligenz* 33, 2 (2019), 181–187.
[43] Per Runeson, Martin Höst, Rainer Austen, and Björn Regnell. 2012. *Case Study Research in Software Engineering - Guidelines and Examples*. John Wiley and Sons Inc.
[44] Safdar Aqeel Safdar, Hong Lu, Tao Yue, and Shaukat Ali. 2017. Mining cross product line rules with multi-objective search and machine learning. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'17)*. Association for Computing Machinery, New York, NY, 1319–1326. https://doi.org/10.1145/3071178.3071261
[45] Marius Schubert, Karsten Kahl, and Matthias Rottmann. 2020. MetaDetect: Uncertainty Quantification and Prediction Quality Estimates for Object Detection. (2020). arXiv:cs.CV/2010.01695.
[46] Claude E. Shannon. 1948. A mathematical theory of communication. *Bell Syst. Tech. J.* 27, 3 (1948), 379–423. http://dblp.uni-trier.de/db/journals/bstj/bstj27.html#Shannon48.

[47] Salman Sherin, Muhammad Uzair khan, and Muhammad Zohaib Iqbal. 2019. A systematic mapping study on testing of machine learning programs. *arXiv e-prints*, Article arXiv:1907.09427 (July 2019), arXiv:1907.09427 pages. arXiv:cs.LG/1907.09427.

[48] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv e-prints*, Article arXiv:1409.1556 (Sept. 2014), arXiv:1409.1556 pages. arXiv:cs.CV/1409.1556.

[49] Lewis Smith and Yarin Gal. 2018. Understanding Measures of Uncertainty for Adversarial Example Detection. (2018). arXiv:stat.ML/1803.08533.

[50] E. Tjoa and C. Guan. 2020. A survey on explainable artificial intelligence (XAI): Toward medical XAI. *IEEE Transactions on Neural Networks and Learning Systems* 32, 11 (2020), 1–21. https://doi.org/10.1109/TNNLS.2020.3027314

[51] Andrew Turpin and Falk Scholer. 2006. User performance versus precision measures for simple search tasks. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'06)*. ACM, New York, NY, 11–18. https://doi.org/10.1145/1148170.1148176

[52] Rosa Arnaldo Valdés, Victor Fernando Gomez Comendador, Javier Alberto Perez Castan, Alvaro Rodriguez Sanz, Luis Perez Sanz, Francisco Javier Saez Nieto, and Eduardo Sanchez Aira. 2019. Bayesian inference in safety compliance assessment under conditions of uncertainty for ANS providers. *Safety Science* 116 (2019), 183–195.

[53] Francesco Verdoja and Ville Kyrki. 2020. Notes on the Behavior of MC Dropout. (2020). arXiv:cs.LG/2008.02627.

[54] Jeroen F. Vranken, Rutger R. van de Leur, Deepak K. Gupta, Luis E. Juarez Orozco, Rutger J. Hassink, Pim van der Harst, Pieter A. Doevendans, Sadaf Gulshad, and René van Es. 2021. Uncertainty estimation for deep learning-based automated analysis of 12-lead electrocardiograms. *European Heart Journal - Digital Health* 2, 3 (2021), 401–415.

[55] Sean Weerakkody, Omur Ozel, Yilin Mo, and Bruno Sinopoli. 2019. Resilient control in cyber-physical systems: Countering uncertainty, constraints, and adversarial behavior. *Foundations and Trends® in Systems and Control* 7, 1–2 (2019), 1–252. https://doi.org/10.1561/2600000018

[56] Michael Weiss and Paolo Tonella. 2021. Fail-safe execution of deep learning based systems through uncertainty monitoring. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST'21)*. 24–35. https://doi.org/10.1109/ICST49551.2021.00015

[57] Michael Weiss and Paolo Tonella. 2021. Uncertainty-wizard: Fast and user-friendly neural network uncertainty quantification. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST'21)*. IEEE, 436–441.

[58] C. S. Wickramasinghe, D. L. Marino, K. Amarasinghe, and M. Manic. 2018. Generalization of deep learning for cyber-physical system security: A survey. In *44th Annual Conference of the IEEE Industrial Electronics Society (IECON'18)*. 745–751. https://doi.org/10.1109/IECON.2018.8591773

[59] Man Zhang, Shaukat Ali, Tao Yue, and Roland Norgre. 2017. Uncertainty-wise evolution of test ready models. *Information and Software Technology* 87 (2017), 140–159.

[60] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. 2019. Uncertainty-wise cyber-physical system test modeling. *Software & Systems Modeling* 18, 2 (2019), 1379–1418.

[61] Man Zhang, Tao Yue, Shaukat Ali, Bran Selic, Oscar Okariz, Roland Norgre, and Karmele Intxausti. 2018. Specifying uncertainty in use case models. *Journal of Systems and Software* 144 (2018), 573–603.

[62] Xiyue Zhang. 2020. Uncertainty-guided testing and robustness enhancement for deep learning systems. In *2020 IEEE/ACM 42nd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion'20)*. 101–103.

[63] Xiyue Zhang, Xiaofei Xie, Lei Ma, Xiaoning Du, Qiang Hu, Yang Liu, Jianjun Zhao, and Meng Sun. 2020. Towards characterizing adversarial defects of deep learning software from the lens of uncertainty. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering (ICSE'20)*. Association for Computing Machinery, New York, NY, 739–751. https://doi.org/10.1145/3377811.3380368

[64] Xin Zhou, Xiaodong Gou, Tingting Huang, and Shunkun Yang. 2018. Review on testing of cyber physical systems: Methods and testbeds. *IEEE Access* 6 (2018), 52179–52194. https://doi.org/10.1109/ACCESS.2018.2869834

Received March 2021; revised November 2021; accepted January 2022