Trust-Based Cloud Machine Learning Model Selection for Industrial IoT and Smart City Services

Basheer Qolomany, *Member, IEEE*, Ihab Mohammed, *Member, IEEE*, Ala Al-Fuqaha, *Senior Member, IEEE*, Mohsen Guizani, *Fellow, IEEE*, and Junaid Qadir, *Senior Member, IEEE*

***Abstract*—With machine learning (ML) services now used in a number of mission-critical human-facing domains, ensuring the integrity and trustworthiness of ML models becomes all important. In this work, we consider the paradigm where cloud service providers collect big data from resource-constrained devices for building ML-based prediction models that are then sent back to be run locally on the intermittently connected resource-constrained devices. Our proposed solution comprises an intelligent polynomial-time heuristic that maximizes the level of trust of ML models by selecting and switching between a subset of the ML models from a superset of models in order to maximize the trustworthiness while respecting the given reconfiguration budget/rate and reducing the cloud communication overhead. We evaluate the performance of our proposed heuristic using two case studies. First, we consider Industrial IoT (IIoT) services, and as a proxy for this setting, we use the turbofan engine degradation simulation data set to predict the remaining useful life of an engine. Our results in this setting show that the trust level of the selected models is 0.49%–3.17% less compared to the results obtained using integer linear programming (ILP). Second, we consider *smart cities* services, and as a proxy of this setting, we use an experimental transportation data set to predict the number of cars. Our results show that the selected model's trust level is 0.7%–2.53% less compared to the results obtained using ILP. We also show that our proposed heuristic achieves an optimal competitive ratio in a polynomial-time approximation scheme for the problem.**

***Index Terms*—Adversarial attacks, automatic model selection, deep learning (DL), Industrial IoT (IIoT), ML as a Service (MLaaS), smart city, trusted machine learning (ML) models.**

I. Introduction

THE GLOBAL market for machine learning (ML) has grown rapidly over the last few years largely due to the fast pace of integrating ML with many facets of everyday life, particularly for enabling smart Internet-of-Things (IoT) services. Most of today's IoT predictive analytics rely on cloud-based services, in which IoT resource-constrained devices continuously send their collected data to the cloud [1]. Resource-constrained devices have limited processing, communication, and/or storage capabilities, and often run on batteries. On the cloud, ML as a Service (MLaaS) providers carry out the prediction process and provide data preprocessing, model training, model evaluation, and model update capabilities [2]. The MLaaS market is expected to exceed $3754 million by 2024 at a compound annual growth rate (CAGR) of 42% in the given forecast period [3]. Typical systems include electrical power grids [4], intelligent transportation and vehicular management [5], health care devices [6], household appliances [7], predictive maintenance (PM) systems [8] in Industrial IoT (IIoT), and many more. However, ML models can be targeted by malicious adversaries [9] due to the participatory nature of such systems. Cyber attacks against critical infrastructure are not just theories, they are very real and have already been used to effect. For example, in December 2015, a cyber attack on Ukraine's power grid left 700 000 people without electricity for several hours [10]. The Stuxnet worm, which was first uncovered in 2010, is believed to be responsible for causing substantial damage to Iran's nuclear program [11]. In March 2016, the U.S. Justice Department indicated that cyber attacks tied to the Iranian regime [12] targeted 46 major financial institutions and a dam outside of New York City.

Perhaps, the most pressing challenge in the emerging cloud computing area is that of establishing trust [13], [14]. Despite the importance of trust in cloud computing, a common conceptual model of trust in cloud computing has not yet been defined [15] and it is becoming increasingly complex for cloud users to distinguish between service providers offering similar types of services in terms of trustworthiness [16]. Trust has been investigated from different disciplinary lenses, such as psychology, sociology, economics, management, and information systems (ISs), but there is no commonly accepted definition of trust [17], [18]. That is, depending on the context, we may think of many different things when someone uses the word "trust." Merriam–Webster's dictionary defines the word "trust" as "assured reliance on the character, ability, strength, or truth of someone or something." Our definition for the trust in this article refers to the ML models that agree most with the predictions of an ensemble of ML models. Therefore, a model

Manuscript received May 20, 2020; revised August 11, 2020; accepted August 28, 2020. Date of publication September 7, 2020; date of current version February 4, 2021. *(Corresponding author: Mohsen Guizani.)*

Basheer Qolomany is with the Department of Cyber Systems, College of Business and Technology, University of Nebraska at Kearney, Kearney, NE 68849 USA (e-mail: qolomanyb@unk.edu).

Ihab Mohammed is with the Department of Computer Science, Western Michigan University, Kalamazoo, MI 49008 USA (e-mail: ihabahmedmoha.mohammed@wmich.edu).

Ala Al-Fuqaha is with the Information and Computing Technology Division, College of Science and Engineering, Hamad Bin Khalifa University, Doha, Qatar, and also with the Computer Science Department, Western Michigan University, Kalamazoo, MI 49008 USA (e-mail: ala@ieee.org).

Mohsen Guizani is with the Computer Science and Engineering Department, Qatar University, Doha, Qatar (e-mail: mguizani@ieee.org).

Junaid Qadir is with the Department of Electrical Engineering, Information Technology University, Lahore 54000, Pakistan (e-mail: junaid.qadir@itu.edu.pk).

Digital Object Identifier 10.1109/JIOT.2020.3022323

that agrees more with the predictions of the ensemble is more "trustworthy" compared to the one that agrees less with the ensemble. For example, assume that we have five models (M_1, M_2, M_3, M_4, and M_5), and the model M_1 agrees with three other models, while M_2 agrees with only two other models, then M_1 is more trustworthy than M_2.

The performance of ML models can be quantified based on their decision time, accuracy, and precision of resulting decisions [19]. However, as such models are used for more critical and sensitive decisions (e.g., whether a drug should be administered to a patient or should an autonomous vehicle stop for pedestrians), it becomes more important to ensure that they provide high accuracy and precision guarantees. Assessing learning models in terms of trustworthiness along with the traditional criteria of decision time, accuracy, and precision establishes a tradeoff between simplicity and power [20]. ML classifiers are vulnerable to adversarial examples, which are samples of input data that are maliciously modified in a way that is intended to cause an ML classifier to misclassify similar examples. Moreover, it is known that adversarial examples generated by one classifier are likely to cause another classifier to make the same mistake [21]. In many cases, the modifications can successfully cause a classifier to make a mistake even though the modifications are imperceptible to a human observer. In general, adversarial attacks can be classified into a misclassification attack or a targeted attack based on its goals [22]–[24]. In a *misclassification attack*, the adversary intends to cause the classifier to output a label different from the original label. In a *targeted attack*, on the other hand, the adversary intends to cause the classifier to output a specific misleading label.

In this article, we envision the paradigm where resource-constrained IoT devices execute ML models locally, without necessarily being always connected to the cloud. Some advantages of our proposed heuristic are its applicability to a number of application scenarios beyond the pale of the traditional paradigm, where it is not desirable to execute the model on the cloud due to latency, connectivity, energy, privacy, and security concerns. Consequently, it is expected that the users should be able to determine the trustworthiness of service providers in order to select them with confidence and with some degree of assurance that their service offerings will not behave unpredictably or maliciously. Our proposed heuristic strives to minimize the communications overhead between the cloud and the resource-constrained devices. Selected ML models are sent to resource-constrained devices to be used. The proposed heuristic also has a limitation that it is not intended to improve the trustworthiness of models trained in federated learning (FL) systems when each client preserves its own data locally. Instead, our approach can be applied to improve the trustworthiness of the centralized approach of learning, when all the clients send their data to an MLaaS provider to build an ML model on the cloud, then this model will be sent back to be hosted on resource-constrained devices. The target of the proposed heuristic is not to handle all different types of attacks. We only consider poisoning attacks on ML classifiers. Within this scenario, an attacker may poison the training data by injecting carefully designed samples to eventually compromise

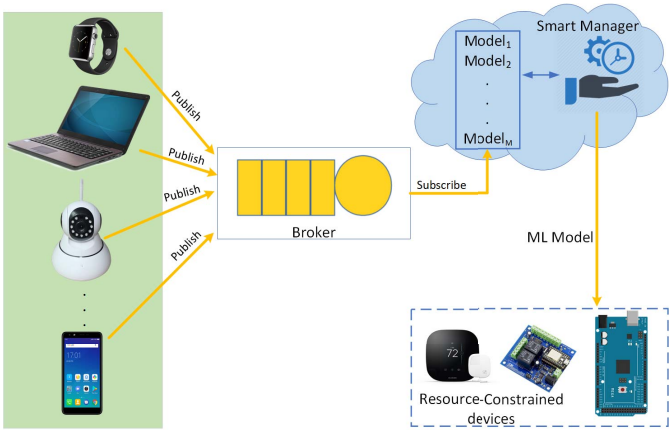

Fig. 1. General architecture for the proposed system of selecting a trustworthy subset of ML models built by different cloud service providers.

the whole learning process. Fig. 1 shows a general architecture for the proposed system. On the cloud side, we have M ML models, $\text{model}_1, \text{model}_2, \ldots, \text{model}_M$.

The main contributions of the work can be summarized as follows.

1) We formulate the problem of finding a subset of ML models that maximize the trustworthiness while respecting the given reconfiguration budget and rate constraints. We also prove the problem is NP-complete.
2) We propose a heuristic that maximizes the level of trust of ML models and finds a near-optimal solution in polynomial time by selecting a subset from a superset of ML models. Our trust metric of an ML model is based on recent and past historical data that measure the degree of agreement of the ML model with other models in an ensemble of ML models.
3) The proposed system has a fail-safe state such that if the proposed heuristic does not find a trusted ML (TML) model in the superset of models, it sends a fail-safe execution alert. This alert informs the resource-constrained devices that no TML model exists in the system. As a result, the resource-constrained devices can fail safely as required by the application that they service.
4) Building on the above insights, we apply the proposed heuristic to two different training data sets. The first data set, based on the CityPulse project [25], is used to predict the number of vehicles as a surrogate use case for smart city services. The second data set, provided by the Prognostics CoE at NASA Ames [26], is used to predict the remaining useful life of a turbofan engine as a surrogate use case for IIoT smart services.
5) We applied the swap x and $100-x$ percentiles approach as a causative adversarial attack by altering the training data set label as we will describe in Section VI.

For the convenience of the readers, Table I provides a list of the acronyms used in this article.

The remainder of this article is organized as follows. Section III presents the most recent related work. The background information of case studies and thread model is introduced in Section II. Section IV discusses the system model

TABLE I
LIST OF IMPORTANT ACRONYMS USED

DL	Deep Learning
FL	Federated Learning
IIoT	Industrial Internet of Things
ILP	Integer Linear Programming
IoT	Internet of Things
IS	Information System
LP	Linear Programming
LSTM	Long Short-Term Memory
ML	Machine Learning
MLaaS	Machine Learning as a Service
NP	Non-deterministic Polynomial-time
PM	Predictive Maintenance
RMSE	Root Mean Square Error
TML	Trusted Machine Learning

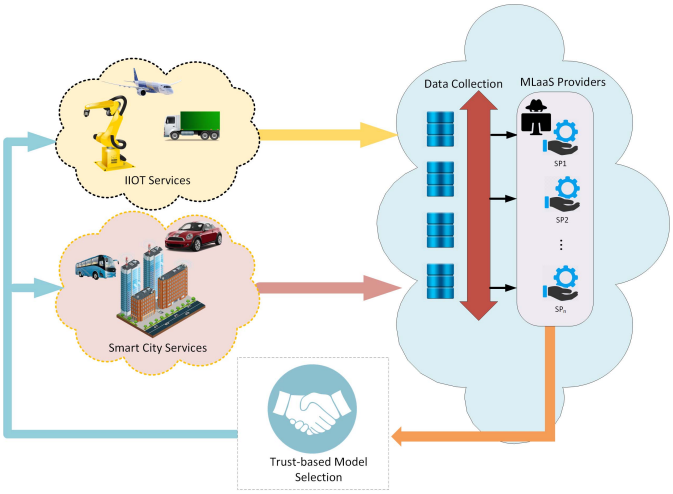

Fig. 2. Trust-based model selection problem for IIoT and smart city case scenarios.

and problem formulation. Section V discusses our proposed solutions and their competitive ratio analysis. Section VI presents our experimental setup, experimental results, and the lessons learned. Finally, Section VII concludes the article and discusses future research directions.

II. BACKGROUND

A. Case Studies

Several case studies could be considered, in which the proposed heuristic helps to gain the best trust level. Here, we discuss two representative case studies. The first case study considers IIoT PM while the second one considers real-time traffic flow prediction in smart cities. Fig. 2 illustrates the trust-based model selection problem addressed in this research and also depicts the two considered case studies. During each decision period, our proposed heuristic switches between the subset of selected models with a goal of maximizing the overall trustworthiness while respecting the given reconfiguration budget and rate.

1) IIoT Predictive Maintenance Case Study: A PM strategy uses ML methods to identify, monitor, and analyze system variables during operation. Also, PM alerts operators to preemptively perform maintenance before a system failure occurs [27]. Being able to stay ahead of equipment shutdowns in a mine, steel mill, or factory, PM can save money and time for a busy enterprise [28]. With PM, the data are collected over time to monitor the state of the machine and are then analyzed to find patterns that can help predict failures. In many cases, it is desirable to have prediction models hosted on resource-constrained embedded devices. PM systems need to provide real-time control of the machines based on the deviation of the real-time flow readings from the predicted ones. In such systems, embedded sensors collect the short-term state of the machine readings, which are relayed to the cloud directly through communications infrastructure or indirectly through the use of ferry nodes. Because of its compute and store capabilities, the cloud is capable of collecting the short-term readings to build long-term big data of sensor readings. These readings are then utilized to build a PM model for each of the underlying flow sensors. The constructed models are then sent back to the flow sensors so that they actuate their associated machines when a deviation is observed between the actual and projected flow readings.

There are scenarios in which cyber attackers attempt to compromise PM models directly. Consequently, the data that leave its internal operating environment is subject to third-party attacks. For instance, an adversary can create a causative attack to poison the learner's classifications. This is possible by altering the training process through influence over the training data. Therefore, when the system is retrained, the learner learns an incorrect decision-making function. Thus, it is important to ensure the trustworthiness of ML models before they are hosted and used on resource-constrained devices.

2) Smart City (Traffic Flow Prediction) Case Study: Traffic flow prediction plays an important role in intelligent transportation management and route guidance. Such predictions can help in relieving traffic congestion, reducing air pollution, and providing secure traffic conditions [29]. Traffic flow prediction heavily depends on historical and real-time traffic data collected from various sensor sources. These sources include inductive loops, radars, cameras, mobile global positioning systems (GPSs), crowdsourcing, social media, etc. Transportation management and control are now becoming more data driven [30]. However, inferring traffic flow under real-world conditions in real time is still a challenging research problem due to the computational complexity of building, training, learning, and storing traffic flow models on resource-constrained devices.

In our proposed approach, various sensor technologies are used to automatically collect short-term data of the traffic flow and send them to the cloud through communications infrastructure or through the use of ferry nodes. The cloud is capable of collecting short-term readings to build long-term big data of sensor readings. These readings are then utilized by MLaaS service providers to build a model. The constructed models are then sent back to be hosted on the resource-constrained devices, in order to predict the traffic flow in real time. Intelligent transportation systems are highly visible, and attacks against them result in a high impact on critical infrastructure. For instance, the attacks can cause vehicular accidents or create traffic jams that affect freight movements, daily commutes, etc. Thus, to make the traffic movement more efficient and improve road safety, road operators need

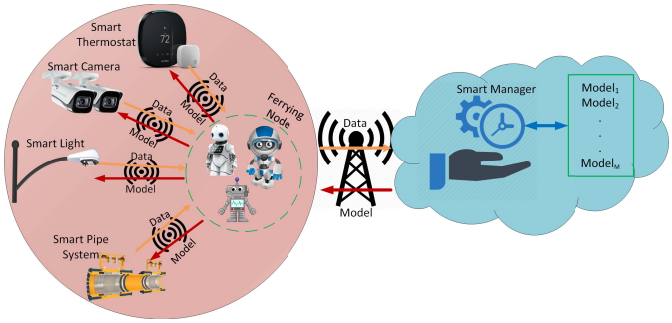

Fig. 3. Exchange of data/models between resource-constrained devices and the cloud using message ferries.

to constantly monitor traffic and current roadway conditions by using an array of cameras and sensors that are strategically placed on the road network. These cameras and sensors send back real-time data to the control center [31]. The data are subject to causative adversarial attacks, which are launched by altering the training process by influencing the training data and consequently causing the learner to learn an incorrect decision-making function.

Fig. 3 illustrates the use of message ferries to collect data from resource-constrained devices. The collected data are delivered to the cloud in order to build ML models by the MLaaS service providers. Next, the ferrying nodes deliver the ML models to be hosted on the resource-constrained devices.

B. Threat Model

1) Adversary Knowledge: For both the case studies, we only consider poisoning attacks on the ML classifiers. Within this scenario, an attacker may poison the training data by injecting carefully designed samples to eventually compromise the whole learning process. Poisoning may thus be regarded as adversarial contamination of the training data. In our experiments, we use the swap x and $100-x$ percentiles attack model as a causative attack against the LSTM algorithm.

2) Adversary Goal: The goal of an adversarial MLaaS provider is to deliver an ML model that results in suboptimal or erroneous results when executed on resource-constrained IoT devices. The incentive of the adversarial MLaaS provider is to seek gains by colluding with business competitors of MLaaS clients.

III. Related Work

In this section, we review recent related works. Fig. 4 shows the research gap that we address in this research. To the best of our knowledge, this article is the first attempt at designing an intelligent polynomial-time heuristic on the cloud that selects ML models that should be hosted on IoT resource-constrained devices in order to maximize the trustworthiness of the overall system.

A. Trust-Based ML Models

Researchers have proposed various approaches to design ML algorithms that are trustworthy when using predictions

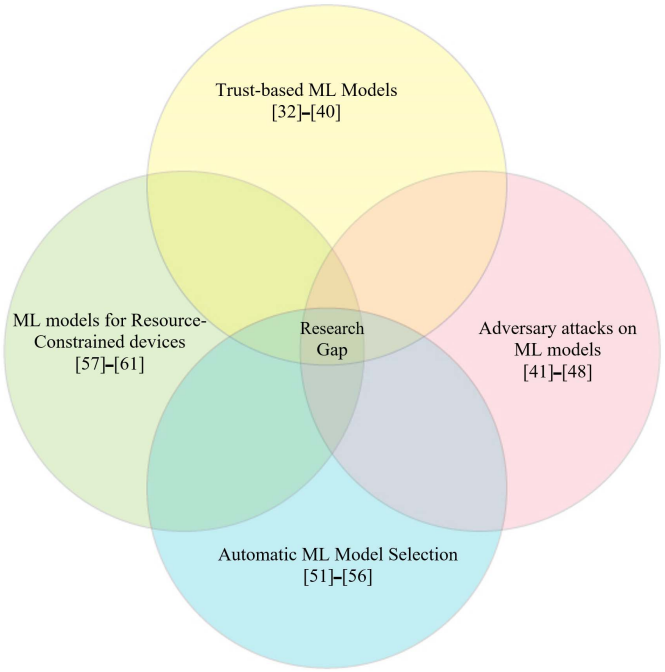

Fig. 4. Research gap addressed in this article.

to make critical decisions in real-world applications, including healthcare, law, self-driving cars, etc. Speicher *et al.* [32] proposed an approach to establish complex ML models by ensuring that in a particular way, a complex model to achieve correct predictions at least on all those data points, where a trusted model was already correct.

Ghosh *et al.* [33] proposed the TML framework for self-driving cars that use principles from formal methods for learning ML models. These ML models satisfy properties in temporal logic by using the model repair or the data from which the model is learned. Zhang *et al.* [34] proposed debugging using trusted items (DUTIs) algorithm that uses trusted items to detect outlier and systematic training set bugs. The approach looks for the smallest set of changes in the training set labels such that the model learned from this corrected training set predicts labels of the trusted items correctly.

Ribeiro *et al.* [35] proposed the LIME algorithm, which explains the predictions of any classifier or regressor in an interpretable manner by approximating an interpretable model locally around the prediction. The authors also proposed a method called SP-LIME to select representative and nonredundant predictions, which provide a global view of the model to users. The authors applied the proposed algorithm on both simulated and human subjects in order to decide between and assess models and also identified reasons for not trusting a classifier. Jayasinghe *et al.* [36] proposed a trust assessment model that specifies the formation of trust from raw data to a final trust value, and they proposed an algorithm based on ML principles that determine whether an incoming interaction is trustworthy, based on several trust features corresponding to an IoT environment. Fariha *et al.* [37] introduced data invariant technique as an approach to achieve TML by reliably detecting tuples on which the prediction of a

machine-learned model should not be trusted. They proposed quantitative semantics to measure the degree of violation of a data invariant and establish that strong data invariants can be constructed from observations with low variance on the given data set. Drozdal *et al.* [38] explored trust in the relationship between human data scientists and models produced by AutoML systems. They find that including transparency features in an AutoML tool increased user trust and understandability in the tool; and out of all the proposed features, model performance metrics and visualizations are the most important information to data scientists when establishing their trust with an AutoML tool.

Wahab *et al.* [39] proposed a solution for maximizing the detection of VM-based DDoS attacks in cloud systems. Their proposed solution has two components. First, they proposed a trust model between the hypervisor and its guest VMs for the purpose of establishing a credible trust relationship between the hypervisor and guest VMs. Second, they designed a trust-based maximin game between DDoS attackers and hypervisors to minimize the cloud system's detection and maximize this minimization under a limited budget of resources. Liu *et al.* [40] made three arguments about the trustworthiness of deep learning (DL) systems to prevent the deception of the algorithm: 1) the trustworthiness should be an essential and mandatory component of a DL system for algorithmic decision making; 2) the trust of a DL model should be evaluated along multiple dimensions in terms of its correctness, accountability, transparency, and resilience; and 3) there should be proactive safeguard mechanisms to enforce the trustworthiness of a DL framework.

In this work, the trust metric of an ML model is based on recent and past historical data that measure the degree of agreement of the ML model with other models in an ensemble of ML models.

B. Adversary Attacks on ML Models

Recent research shows that ML models trained entirely on private data are still vulnerable to adversarial examples, which have been maliciously altered so as to be misclassified by a target model while appearing unaltered to the human eye [21], [22]. Madry *et al.* [41] proposed an approach to study the adversarial robustness of neural networks through the lens of robust optimization, and this approach enables to identify methods for both training and attacking neural networks models. Finlayson *et al.* [42] demonstrated that adversarial examples are capable of manipulating DL systems. They synthesize a body of knowledge about the healthcare system across three clinical domains to argue that medicine may be uniquely susceptible to adversarial attacks. Huang *et al.* [43] discussed the effective ML techniques against an adversarial opponent. They introduce two ML models for modeling an adversary's capabilities and discuss how specific application domains, features, and data distribution restrict an adversary's attacks. Saadatpanah *et al.* [44] discussed how the ML methods in industrial copyright detection systems are susceptible to adversarial attacks and why those methods are particularly vulnerable to attacks. Ren *et al.* [45] introduced the theoretical foundations, algorithms, and applications of adversarial attack and defense techniques in DL models. Chakraborty *et al.* [46] provided a discussion on different types of adversarial attacks with various threat models and also elaborated on the efficiency and challenges of recent countermeasures against them. Akhtar and Mian [47] presented a comprehensive survey paper on adversarial attacks on DL in computer vision. Yuan *et al.* [48] investigated and summarized the approaches for generating adversarial examples, applications for adversarial examples, and corresponding countermeasures for deep neural network models.

C. Automatic Model Selection

Researchers have proposed various automatic selection methods for ML algorithms. ML model selection is the problem of determining which algorithm, among a set of ML algorithms, is the best suited to the data [49]. Choosing the right technique is a crucial task that directly impacts the quality of predictions. However, deciding which ML technique is well suited for processing specific data is not an easy task, even for an expert, as the number of choices is usually very large [50].

Auto-WEKA [51] considers all 39 ML classification algorithms implemented in Weka to automatically and simultaneously choose a learning algorithm. Auto-WEKA uses sequential model-based optimization and a random forest regression model to approximate the dependence of a model's accuracy on the algorithm and hyperparameter values. Using an approach similar to that in Auto-WEKA, Komer *et al.* [52] developed the software *hyperopt-sklearn*, which automatically selects ML algorithms and the hyperparameter values for Scikit-learn.

In another work, Sparks *et al.* [53] proposed MLbase, an architecture for automatically selecting ML algorithms that supports distributed computing on a cluster of computers by combining better model search methods, bandit methods, batching techniques, and a cost-based cluster sizing estimator.

Lokuciejewski *et al.* [54] presented a generic framework for automatically selecting an appropriate ML algorithm for the compiler generation of optimization heuristics. Leite *et al.* [55] proposed a method called active testing for automatically selecting ML algorithms, which exploits metadata information concerning past evaluation results to recommend the best algorithm using a limited number of tests on the new data set.

Van Rijn *et al.* [56] proposed a method for automatically selecting algorithms. They addressed the problem of algorithm selection under a budget, where multiple algorithms can be run on the full data set until the budget expires. Their method produces a ranking of classifiers and takes into account the run times of classifiers.

D. ML Models for Resource-Constrained Devices

Researchers have worked on the inference problem on tiny resource-constrained IoT devices, which are not necessarily always connected to the cloud. Kumar *et al.* [57] and Gupta *et al.* [58] developed tree and k-nearest neighbor-based algorithms, called Bonsai and ProtoNN, respectively, for classification, regression, ranking, and other common IoT tasks.

Their algorithm can be trained on the cloud and then be hosted onto resource-constrained IoT devices based on the Arduino Uno board. Bonsai and ProtoNN maintain prediction accuracy while minimizing model size and prediction costs. Motamedi *et al.* [59] presented a framework for the synthesis of efficient convolutional neural networks (CNNs) inference software targeting mobile system on chip (SoC)-based platforms. They used parallelization approaches for deploying a CNN on SoC-based platforms. Meng *et al.* [60] presented two-bit networks (TBNs) approach for CNN model compression to reduce memory usage and improve computational efficiency in terms of classification accuracy on resource-constrained devices. They utilized parameter quantization for computation workload reduction. Shoeb and Guttag [61] presented an ML approach on a wearable device to identify epileptic seizures through analysis of the scalp electroencephalogram, a noninvasive measure of the brain's electrical activity.

IV. System Model and Problem Formulation

In this article, we assume an MLaaS provider that has M ML models from which a subset needs to be selected and deployed on IoT devices for T time slots. P is a constant matrix of size $M \times T$, where element $p_{i,j}$ indicates the trust value obtained by model i at time j. This matrix is created based on recent and past historical data that measure the degree of agreement of ML model i with the other $M-1$ models in the ensemble of ML models. B is the maximum number of allowed ML model reconfigurations during T time slots. A is a variable matrix of size $M \times T$, where element $a_{i,j} \in \{0, 1\}$. $a_{i,j} = 1$ indicates that the model i at time j is trustworthy; and $a_{i,j}$ is equal to zero, otherwise. In other words, A is a variable selection matrix where a value of 1 in row r column c indicates that model number r is selected at time c; otherwise, if the value is 0, then model r is not selected at time slot c. Thus, the objective of the formulation is to find the values of $a_{i,j}$ and $p_{i,j}$ such that the selected models maximize the overall trust values during the entire time period as shown in (2).

As our proposed heuristic depends on the prediction output of the ML model, it can be used with any supervised ML algorithms as classification and regression problems. In our experiments, we use the proposed approach to select trusted LSTM models in regression problems. To compute the trust level of model i at time j, we use (1), which assigns a higher trust metric to models that agree more with the average of all models. This equation is inspired by the majority voting approach presented in the literature to quantify the level of trust [62]–[66]. Therefore, model i at time j is assigned trust level $p_{i,j}$ that represents the degree of agreement [i.e., the reciprocal of the degree of deviation $([\sum_{k=1}^{M} d(O_{i,j}, O_{k,j})]/M)$] of model i with other models in the ensemble of ML models. $d(O_{i,j}, O_{k,j})$ is a function that provides the distance between $O_{i,j}$ and $O_{k,j}$. The trust-level metric ranges from 0 to $p_{\max}$, where a higher value indicates a higher level of trust

$$p_{i,j} = \min\left(p_{\max}, \left[\frac{\sum_{k=1}^{M} d\big(O_{i,j}, O_{k,j}\big)}{M}\right]^{-1}\right) \tag{1}$$

TABLE II
Description of Formulation Parameters

Parameter	Meaning
A	Variable matrix of size M×T, $a_{i,j} \in \{0, 1\}$
B	Maximum number of allowed ML model configurations
H	Constant value which represents the threshold of maximum trust level value selected from the fractional solution
ϵ	Constant small value that is subtracted from H value
M	Number of ML models
$O_{i,j}$	Output of the model i at time j
P	Constant matrix of size M×T, which represents the trust value obtained by all models at all time slots
$P_{i,j}$	Trust value obtained by model i at time j
P_{Max}	Maximum attainable trust level
R	Maximum rate of reconfiguration
T	The number of time slots

where $p_{\max}$ is the maximum attainable trust level in the given application domain and $p_{i,j}$ is the trust level of model i at time j, $O_{i,j}$ is the output of model i at time j, and $O_{i,j} \in \mathbb{R}$.

Problem Formulation: The goal of this work is to maximize the trust level gained by selecting a subset of ML models from a superset of models to be hosted on resource-constrained devices for a period of time R, where $0 \leq R \leq T$. The number of reconfigurations is limited to B and the maximum rate of reconfiguration is limited to R. We formulate the problem using integer linear programming (ILP) as follows:

$$\max \sum_{j=1}^{T}\sum_{i=1}^{M} a_{i,j} \cdot p_{i,j} \tag{2}$$

$$\text{s.t.} \sum_{i=1}^{M} a_{i,j} = 1 \qquad \forall j \in 1 \ldots T \tag{3}$$

$$a_{i,j} \in \{0, 1\} \qquad \forall i \in 1 \ldots M$$

$$\forall j \in 1 \ldots T \tag{4}$$

$$\frac{1}{2} \cdot \sum_{i=1}^{M}\sum_{j=2}^{T} |a_{i,j} - a_{i,j-1}| \leq B \tag{5}$$

$$\frac{1}{2} \cdot \sum_{i=1}^{M}\sum_{j=k}^{k+\lceil \frac{T}{B} \rceil} |a_{i,j} - a_{i,j-1}| \leq R$$

$$\forall k \in 1 \ldots \left(T - \frac{T}{B}\right). \tag{6}$$

The first constraint in (3) is to ensure that only one ML model is selected at each time slot because there will be only one ML model hosted in a resource-constrained device at a time. The second constraint in (4) indicates that this formulation is combinatorial, where the values can either be 0 or 1 with 1 indicating the trustworthiness of the ML model and 0 indicating that it is not. In order to comply with the maximum number of allowed reconfigurations (B), the third constraint in (5) is used. The fourth constraint in (6) restricts the solution to adhere to the models' maximum reconfiguration rate R (i.e., the maximum number of reconfiguration per time unit). Table II summarizes the description of the formulation parameters.

Input Matrix

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
M_1	1	0	0	1	0	1	0	1	0	0	1	0	1	1	0	0
M_2	1	1	0	1	1	0	1	1	1	0	0	0	0	1	0	0
M_3	1	1	1	1	0	0	0	1	0	1	0	0	0	1	0	1
M_4	0	1	1	1	0	1	0	0	1	0	1	0	1	0	1	1
M_5	1	1	0	0	1	0	1	1	0	0	1	0	1	0	1	0

Output Matrix

	T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}	T_{13}	T_{14}	T_{15}	T_{16}
M_1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
M_2	0	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0
M_3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
M_4	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1	1
M_5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. 5. Illustration of our proposed splice heuristic work. Here, the splice heuristic selects three longest consecutive sequences of 1-s segments, then merges the adjacent unselected segments.

V. Proposed Solution

In this section, we discuss the proposed algorithms for the lower bound and competitive solution along with the upper bound algorithm. We also illustrate the proof of NP-completeness of selecting a subset of ML models from a superset of ML models in order to maximize the trust level of ML models.

A. Lower Bound

To find a lower bound solution, we propose the *splice heuristic* shown in Algorithm 1. The heuristic accepts A, a matrix of size $M \times T$, where the element $a_{i,j}$ represents the trust level of model i at time slot j. Initially, the heuristic considers A as one unselected segment. Next, the heuristic iteratively uses three steps. In the first step, for each unselected segment that is at least R in length, the heuristic finds the model (row) k with the longest consecutive sequence of 1 s (i.e., the highest trust level). In the second step, the segment that has the highest trust level is marked as selected. Additionally, the row k is selected by setting all the values in row k to 1 and in rows other than k to 0. The third step merges adjacent selected segments (from the previous rounds) into a single selected segment if they share the same selected model.

These three steps are repeated until at most B segments are selected or on unselected segments are left. Finally, the heuristic identifies unselected segments if such segments exist. For each unselected segment, the heuristic finds the trust level using the highest trust level from a selected adjacent segment, if one exists. Finally, the heuristic compares the trust level resulting from the adjacent ML models (if they exist) and chooses the one with the highest trust level.

The example in Fig. 5 illustrates the details of our proposed splice heuristic. In this example, we assume that $R = 4$ and $B = 2$. Consequently, the heuristic selects $B+1 = 3$ segments that maximize the trust level. The first section covers time slots $T_1 - T_4$ with the selected ML model M_3. M_2 is selected in the second segment, which covers the time slots $T_7 - T_{10}$. Finally, the last segment has M_4 selected in the time slots $T_{13} - T_{16}$. After that, the heuristic determines which ML model to use for the remaining unselected segments. For the time slots $T_5 - T_6$, M_2 is selected based on the selected adjacent segment to the right. In addition, for the time slots $T_{11} - T_{12}$,

Algorithm 1 Splice Heuristic to Find a Lower Bound Solution

Input: Matrix A of size $M \times T$ where element $a_{i,j}$ represents the trust level of model i at time slot j;
Maximum number of allowed reconfigurations B;
Maximum reconfiguration rate R.
Output: matrix A, with each column having only 1 entry to indicate the selected ML model at the given time slot.

Mark A as one unselected segment
Set $i = 0$
while $i \leq B$ AND number of unselected segments > 0 **do**
  Set *flag* = False
  Identify unselected segment j with at least R columns that has the longest consecutive sequence of 1s in row k.
  if Segment j exists **then**
    Set all entries of row k to 1, and set all entries of other rows of segment j to 0
    Mark segment j as selected
    if w is a selected segment that is adjacent to j and both have 1s in the same row **then**
      Merge segments w and j
      Set *flag* = True
    end if
  end if
  if *flag* = False **then**
    Set $i = i + 1$
  end if
end while
Merge adjacent unselected segments into one
for every unselected segment j **do**
  Set *leftSum* = 0, *rightSum* = 0, *selectedRow* = 0
  if there is a selected segment w with selected row k left adjacent to segment j **then**
    Set *leftSum* = sum of values of row k in segment j
    Set *selectedRow* = k
  end if
  if there is a selected segment w with selected row z right adjacent to segment j **then**
    Set *rightSum* = sum of values of row z in segment j
    if *rightSum* > *leftSum* **then**
      Set *selectedRow* = z
    end if
  end if
  Set all entries of *selectedRow* of segment j to 1 and all entries of the other rows to 0
end for
Return A as the best solution.

Algorithm 2 Fixing Heuristic to Produce a Competitive Solution

Input: Matrix A of size $M \times T$ where element $a_{i,j}$ represents the trust level of model i at time slot j;
Maximum number of allowed reconfigurations B;
Maximum reconfiguration rate R;
Maximum trust level selected from fractional solution H;
Epsilon ϵ, a small value subtracted from H.

Output: matrix A with each column having only one value as 1 to indicate the selected ML model at the given time slot.

PART I - Fixing

Set $XSplice = A$
Run the Splice heuristic on matrix *XSplice*
Set *PreviousTrustLevel* = element-wise sum of A & *XSplice* where & is the bitwise AND operator
Set $XFraction = A$ and apply linear programming to generate a fractional solution
Set $X = XFraction$
Compute *CurrentTrustLevel* using PART II
while $H > 0$ AND equation 1 through 3 are satisfied AND *CurrentTrustLevel* > *PreviousTrsutLevel* **do**
    Set *PreviousTrustLevel* = *CurrentTrustLevel*
    Set $H = H - eps$
    Set $X = XFraction$
    Compute *CurrentTrustLevel* using PART II
end while
Return X as the best solution.

PART II - Computing CurrentTrustLevel

Set *rowNum* = -1
for t from t_0 to T **do**
    if maximum value from column t of matrix $X \geq$ H **then**
        Set this maximum value to 1 and set rest of values in column t to 0
        Set *rowNum* = row number of the maximum value
    else if $rowNum = -1$ **then**
        Set the maximum value to 1 and set rest of values in column t to 0
    else
        Set the value at *rowNum* to 1 and set rest of values in column t to 0
    end if
end for
Set *CurrentTrustLevel* = element-wise sum of A & X where & is the bitwise AND operator

M_4 is selected based on the selected adjacent segment to the right.

B. Upper Bound

We relax the ILP formulation presented in Section IV to a linear programming (LP) problem by replacing the constraint (4) with

$$a_{i,j} \in [0, 1] \qquad \forall i \in 1 \ldots M. \tag{7}$$

This relaxed formulation produces an upper bound solution for our problem.

C. Competitive Solution

To produce a competitive solution, we propose the *fixing heuristic* shown in Algorithm 2. The algorithm accepts the matrix A, of dimensions $M \times T$, where the element $a_{i,j}$ represents the trust level of model i at time slot j. The heuristic selects a maximum of $B + 1$ ML models (which results in a maximum of B model reconfigurations) to be used during T in order to maximize the overall trust level. The proposed heuristic employs two constants: 1) a threshold H that represents the maximum trust level selected from the fractional solution ($0 < H < 1$) and 2) epsilon ϵ, which is a small value that is subtracted from the value of H during each iteration of the fixing process ($0 < \epsilon < 0.1$).

The proposed heuristic finds the lower bound solution first using the splice heuristic (Algorithm 1) on matrix A. Next, the proposed fixing heuristic applies LP on matrix A to find a fractional upper bound solution using H. Actually, the ML model with the highest trust level in each time slot of A is compared with H. The highest trust level is rounded to 1 if it is greater than or equal to H while setting all other ML models to 0 during that time slot. The same process is applied for trust levels less than H. If the highest trust level is less than H in any time slot, the selected ML model in the previous time slot is selected for this time slot and is rounded to 1 while other ML models are set to 0. After converting the matrix into a binary one (i.e., 0 or 1 entries), the upper bound solution is computed by counting the number of entries in A that are set to 1. If the upper bound solution is found to be greater than the lower bound solution, the lower bound solution is set to the value of the upper bound solution. Also, H is reduced by ϵ and the upper bound solution is recomputed in the hope of finding a better solution. This process is repeated as long as the upper bound solution is improved.

Because our proposed algorithm depends on the solution produced by LP, which can be solved using the Simplex algorithm, then the complexity of our proposed algorithm is similar to the complexity of the Simplex algorithm that has polynomial-time complexity under various probability distributions.

D. Proof of NP-Completeness

In this section, we show that the problem discussed in this article can be reduced from the decision version of the set cover problem, which is known to be NP-complete.

We define the universe $\mathcal{U}$ as a set of tuples $(i, j), i, j \in T$ and $i \leq j$. Each tuple (i, j) represents a time interval that starts at time i and ends at time j during which the system uses the same model without any reconfigurations. We also define $\mathcal{S}$ as a family of subsets of $\mathcal{U}$. The union of $\mathcal{S}$ results in a period that covers $\mathcal{U}$. In other words, the union of $\mathcal{S}$ results in a period that starts at time 0 and ends at time T. Now, the cardinality of S is represented as follows:

$$0 \leq ||S|| \leq \left[\sum_{i=1}^{T} \binom{T}{i} \right] * M. \tag{8}$$

If k represents the maximum number of model reconfigurations, the objective of our problem is to find k subsets from $\mathcal{S}$ while maximizing the total trust level. This problem is similar to the decision version of the set cover problem. The universe $\mathcal{U}$ and the set $\mathcal{S}$ of our problem are the same as the universe $\mathcal{U}$ and set $\mathcal{S}$ in the set cover problem. However, in our problem, every element is a tuple. The maximum number of model reconfigurations k is the same as the integer number k in the set cover problem. Consequently, the problem introduced in this article is NP-complete.

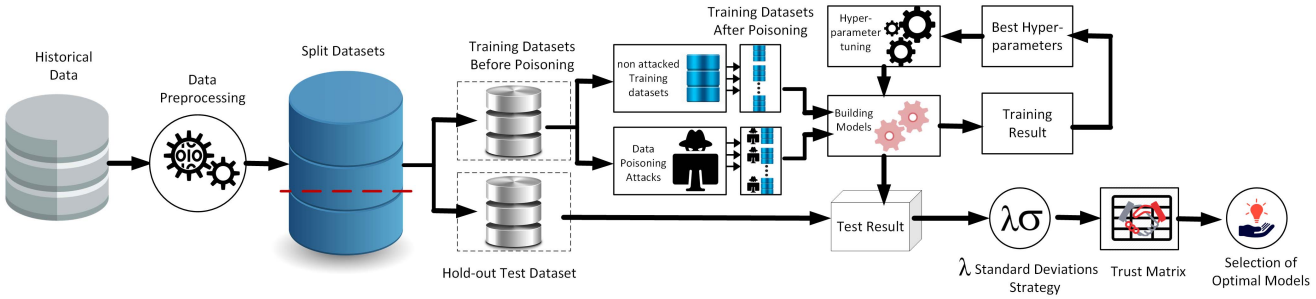

Fig. 6. The data processing pipeline utilized in our experimental studies starting from the data collection phase and ending with the selection of trustworthy ML models.

E. Worst Case Analysis (Competitive Ratio Analysis)

The performance of our proposed fixing heuristic is at least as good as that of the splice heuristic. Consequently, the worst case scenario is encountered when the proposed heuristic performs as the splice heuristic. When the maximum number of allowed reconfigurations is set to B, the splice heuristic finds $(B+1)$ segments, each of length R that provide the maximal trust level.

Proposition 1: For any configuration X, the splice heuristic's worst case performance has a competitive ratio of $O(1)$ when R is proportional to T and B is constant.

Proof: Let ALG be the splice heuristic and OPT be the optimal algorithm. The first part of the splice heuristic (Algorithm 1), specifically steps 1 through 17, finds the segments with the longest consecutive sequence of 1 s. Actually, both ALG and OPT select those segments since they have the largest sum of values (i.e., maximum trust levels). Specifically, those segments have a total length of $R(B+1)$. However, the two approaches differ in the rest of the solution, which is the unselected segments in ALG. Now, at the end of the first part and in the worst case scenario, Algorithm 1 may already have performed B reconfigurations and cannot use more reconfigurations. In other words, for every unselected segment, Algorithm 1 can only use either of the selected models in the adjacent selected segments but never a different model. Consequently, in the worst case scenario, the two models in the selected segments adjacent to the unselected segment are different. Thus, the second part of Algorithm 1, specifically steps 17 through 32, will pick the model that has the largest sum in the unselected segment. In the worst case scenario, both the left and right adjacent selected segments may have the same value when both used in the unselected segment, and therefore, Algorithm 1's maximum loss is half the segment length. However, the loss can never be less than half the segment length. Mathematically, in the second part of the solution, OPT achieves a maximum of $T-R(B+1)$ while ALG achieves a minimum of $(1/2)[T-R(B+1)]$. Consequently, the following proof is concluded as follows.

Competitive Ratio = [ALG(X)/OPT(X)]

$$
\begin{aligned}
&= \frac{R(B+1)+\frac{1}{2}[T-R(B+1)]}{R(B+1)+[T-R(B+1)]} \\
&= \frac{\frac{1}{2}T+\frac{1}{2}R(B+1)}{T} \\
&= \frac{1}{2}\frac{T+R(B+1)}{T} \\
&= \frac{1}{2}\left[\frac{T}{T}+\frac{RB}{T}+\frac{R}{T}\right] \\
&= O\left(\frac{R(B+1)}{T}\right).
\end{aligned}
$$

This competitive ratio is $O(1)$ when R is proportional to T and B is constant. ■

VI. Performance Evaluation

In order to evaluate the performance of the proposed heuristic, we designed and implemented the data processing shown in Fig. 6. In our experiments, we focused on two case studies that serve as proxies for smart city and IIoT services. In our experiments, we trained multiple ML models using sampled experimental data sets to simulate multiple service providers sending ML models to resource-constrained devices.

A. Experimental Setup

The first case study is a proxy for smart city services in which the City Pulse EU FP7 project [25] data set is used for traffic prediction. This data set conveys the vehicular traffic volume collected from the city of Aarhus, Denmark, observed between two points for a set duration of time over a period of six months.

The second case study is a proxy for IIoT services in which the Turbofan engine degradation simulation data set, provided by the Prognostics CoE at NASA Ames [26], is used for predicting the remaining useful life of engines. Engine degradation simulation was carried out using a C-MAPSS tool. The goal is to predict the remaining useful life or the remaining number of cycles before the turbofan engine reaches a level that no longer performs up to requirements. The requirement is based on data collected from sensors located on the turbofan and also on the number of cycles completed. The prediction helps to plan maintenance in advance. The training data consists of multiple multivariate time series with "cycle" as the time unit, together with 21 sensor readings for each cycle. Each time series can be assumed as being generated from a different engine of the same type. The testing data have the same data schema as the training data. The only difference is that the data does not indicate when the failure occurs. Finally, the ground-truth data provide the number of remaining work

TABLE III
DESCRIPTION OF DATA SETS FOR CASE STUDIES

Dataset	Number of records in each training sample	Number of records in testing set	Number of features
Traffic volume	One week of hourly counting the number of vehicles	Seven weeks of counting the number of vehicles	12 lags of number of vehicles
Turbofan engine degradation	3716 records sampled from 250 engines	1800 records sampled from 250 engines	27 features include engine id, cycle number, 3 settings, 21 sensors readings, and remaining useful life (RUL)

cycles for the engines in the test data. Table III shows the description of data sets for both use case studies.

Each data set is divided into training and testing subsets. Each training data set is sampled into 27 different data sets that we used to train 27 deep LSTM models (17 of those models are benign and 10 are malicious, and 20% of the training data of the malicious models are poisoned with causative attacks). Even though it is possible to sample our experimental data sets differently to produce a higher/lower number of ML models, our choice of 27 was based on exploratory experiments designed to explore the maximum number of ML models that can be produced from our experimental data sets without affecting the accuracy of the generated models. Specifically, we use the swap x and $100-x$ percentiles attack model as a causative attack to intentionally poison the learners' classifications by altering the labels of the training data set. In the swap x and $100-x$ percentiles attack, the x percentile value is exchanged with the $100-x$ percentile value. As an example of the swap x and $100-x$ percentiles attack, consider the numeric data set in Fig. 7. To find the ith percentile, we need to sort the values in the unsorted list in ascending order. Next, we multiply i% by the total number of items in the list (i.e., ten items). Now, for example, let us find 20th and 80th percentiles in the list. 20th percentile = $0.2 \times 10 = 2$ (item index), which is value 174 in the list. 80th percentile = $0.8 \times 10 = 8$ (item index), which is value 188 in the list. Now, to swap the x and $100-x$ percentiles in this data set, every 174 will be replaced with 188, and every 188 will be replaced with 174 in the region in which we want to introduce the swap x and $100-x$ percentiles attack.

Since our goal in this article is to assess the trust level of LSTM models, we used grid search to tune the number of hidden layers, the number of neurons in a layer, the batch size, and the activation function parameters that play a major role in the building of LSTM models [67]–[69]. ML models are trained using different configurations. Each configuration includes different values for the number of hidden layers, the number of neurons in each layer, and activation functions. Finally, we select the configuration that gives the best accuracy. Table IV shows the ranges of the configuration parameters used in our experiments to generate the LSTM models.

After building the models, we evaluated our model selection approach on two experimental data sets. Every row in the traffic data set represents the number of vehicles during a specific hour. On the other hand, a row in the Turbofan engine

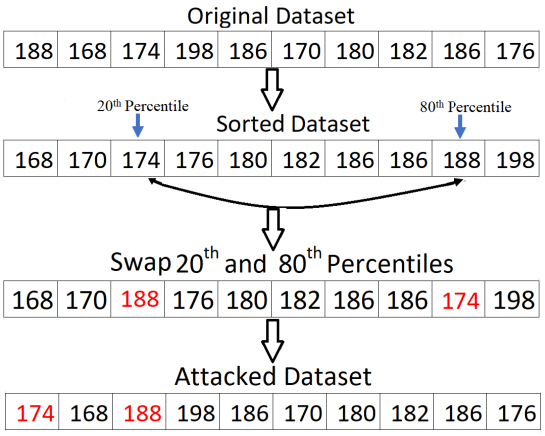

Fig. 7. Example of swap x and $100-x$ percentiles attack model.

TABLE IV
CONFIGURATION PARAMETER RANGES

Parameter	Value
Number of hidden layers	[1–6]
Number of Neurons	[4–1024]
Activation function	Rectified Linear Unit (ReLU)
Batch size	[72–200]
Epochs	[10-50]

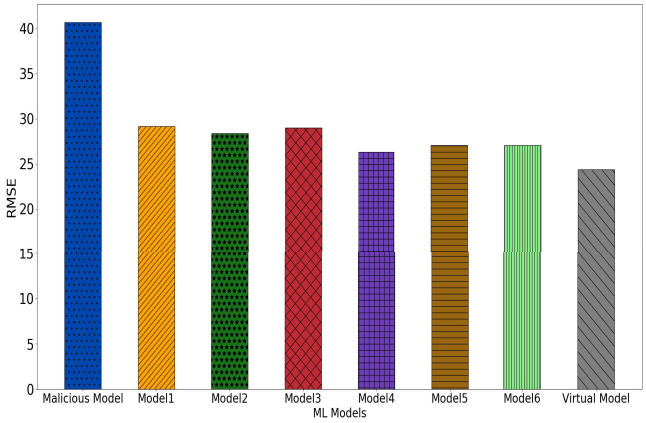

Fig. 8. Smart city traffic flow prediction use case: RMSE of the models using the fixing heuristic versus individual models.

data set represents the remaining useful life during a specific cycle. Figs. 9–12 for the smart city traffic flow prediction use case and Figs. 14–17 for the IIoT PM use case show that the trust level varies between the two data sets. This is because the number of observations in the test set is different for the two experimental data sets. For the traffic data set, the number of observations is ∼900 while for the turbofan data set, the number of observations is ∼1800. Next, we utilize λ standard deviations strategy, which is inspired by the six sigma strategy [70] to exclude the malicious models by identifying and removing the causes of defects and minimizing variability using statistical methods [namely, the mean and the standard deviation as shown in (9) and (10)], which leads to better trust prediction models

$$\text{OutUpper} = \mu + \lambda \times \sigma \tag{9}$$

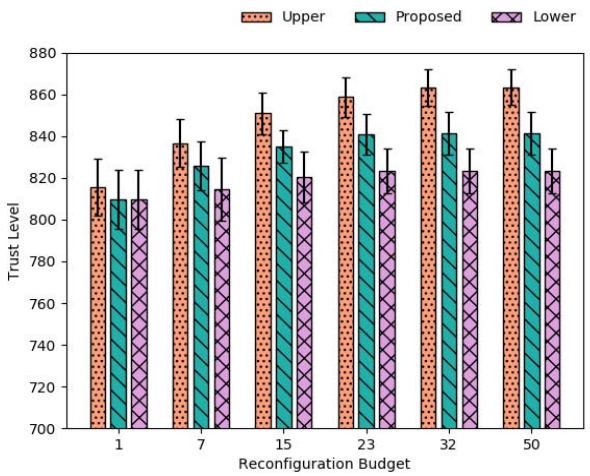

Fig. 9. Smart city traffic flow prediction use case: trust level of upper bound, lower bound, and proposed heuristics.

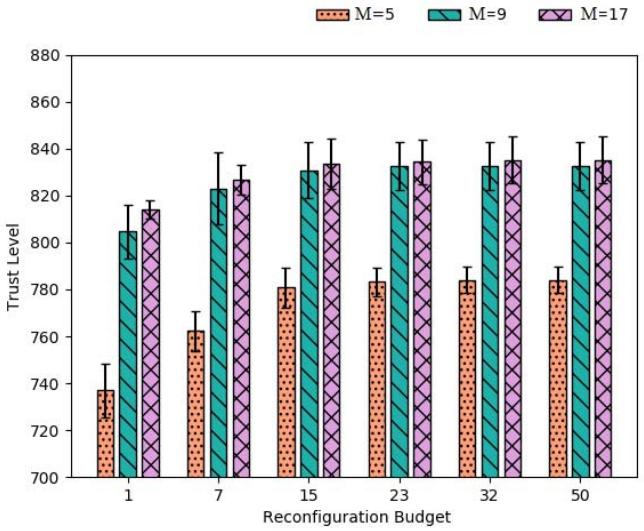

Fig. 10. Smart city traffic flow prediction use case: the effect of the number of selected models on the trust level.

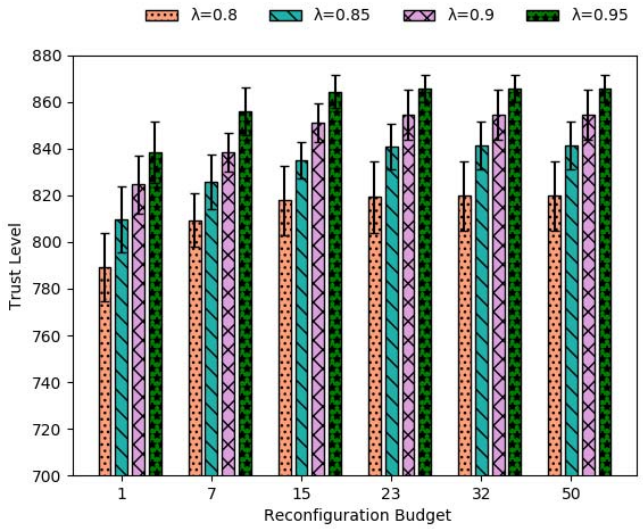

Fig. 11. Smart city traffic flow prediction use case: the effect of λ on the trust level.

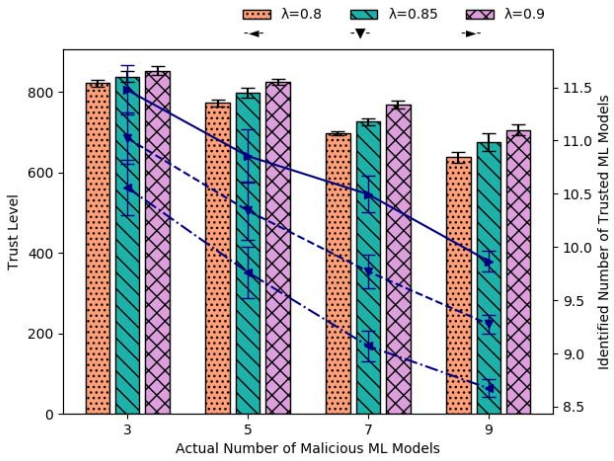

Fig. 12. Smart city traffic flow prediction use case: the effect of malicious models on the trust level.

$$\text{OutLower} = \mu - \lambda \times \sigma. \tag{10}$$

Every time step, we compute μ, which is the mean of the outputs of all models. Also, we compute σ, the standard deviation of the outputs of all models. λ defines the model exclusion strategy (i.e., any model that has an output that is $> \mu + \lambda \times \sigma$ or is $< \mu - \lambda \times \sigma$ is excluded). The λ standard deviations strategy produces a trust matrix of size $M \times T$, with 1 indicating a trusted model and 0 indicating a malicious model. The resulting matrix is then used as the input (i.e., matrix A) for the proposed fixing heuristic.

B. Experimental Results

In this section, we discuss the results of using the proposed fixing heuristic along with the lower bound and upper bound heuristics on the two data sets introduced in the previous section.

1) Traffic Flow Volume Prediction: In our first experiment, we studied the root-mean-square error (RMSE) of the models selected using our proposed fixing heuristic vis-à-vis the individual models. We set the reconfiguration budget B to 7 as shown in Fig. 8. As the figure shows, the proposed fixing heuristic results in 11%–66.95% less RMSE when compared to the individual models.

In our second experiment, the trust levels resulting from the three heuristics are compared under different reconfiguration budgets as illustrated in Fig. 9. The figure shows the confidence interval for five replications. In each replication, the malicious model is applied on a different model (e.g., MM_1, $MM_2, \ldots,$ or MM_n). In this experiment, we set λ to 0.85, M to 7, and the number of malicious models C to 1. The number of nonmalicious models is $M - C$.

In our third experiment, the trust level of the selected models is studied as the number of models M is varied (5, 9, and 17) as illustrated in Fig. 10. In this experiment, we set λ to 0.85 and

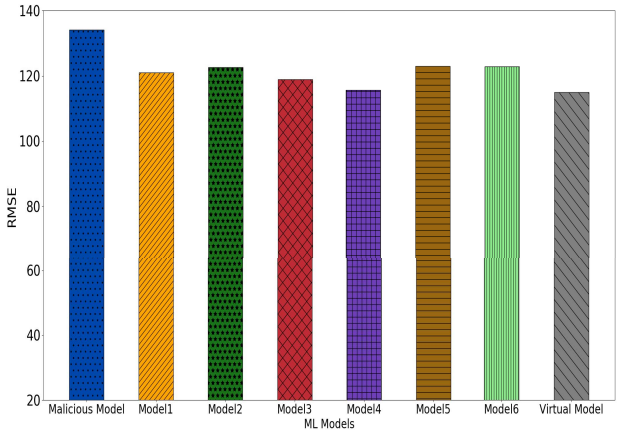

Fig. 13. IIoT PM use case: RMSE using the fixing heuristic versus individual models.

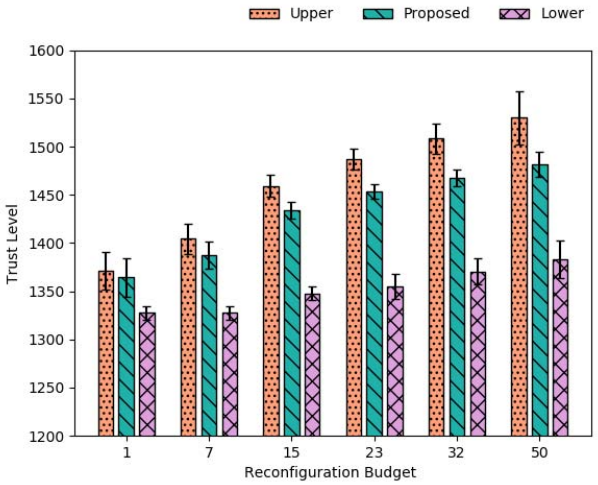

Fig. 14. IIoT PM use case: trust level of upper bound, lower bound, and proposed heuristics.

C to 1. In addition, the figure indicates that as M is increased, the trust level of the selected models is increased too.

Fig. 11 shows the results of our fourth experiment. In this experiment, the effect of using different values of λ (0.8, 0.85, 0.9, and 0.95) on the trust level of the selected models is analyzed. In this experiment, we set C to 1 and M to 7. The figure shows this effect for different reconfiguration budgets B. The figure indicates that as λ is increased, the trust level of the selected models is increased too.

Fig. 12 shows the effect of the number of the malicious LSTM models C (3, 5, and 7) on the trust level of the selected models for different values of λ (0.8, 0.85, and 0.9). The figure also shows the actual number of malicious LSTM models versus the identified number of malicious LSTM models. In this experiment, we set M to 17 and B to 7.

2) Predictive Maintenance in IIoT: In our first experiment, we studied the RMSE of the models selected using our proposed fixing heuristic vis-à-vis the individual models. We set the reconfiguration budget B to 7 as shown in Fig. 13. As the figure shows, the proposed fixing heuristic results in 0.5%–15% less RMSE when compared to the individual models.

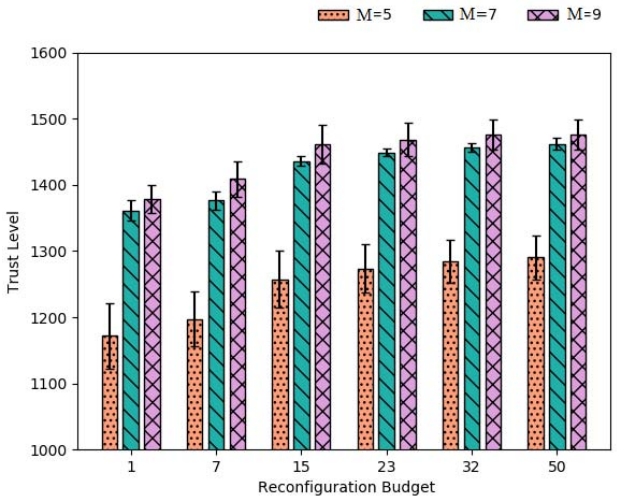

Fig. 15. IIoT PM use case: the effect of the number of selected models on the trust level.

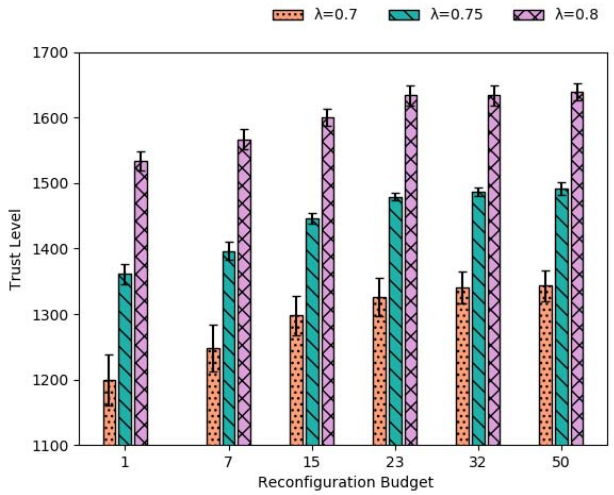

Fig. 16. IIoT PM use case: the effect of λ on the trust level.

In our second experiment, the trust levels resulting from the three heuristics are compared under different reconfiguration budgets as illustrated in Fig. 14. The figure shows the confidence interval for five different replications. In each replication, the malicious model is applied to a different model (e.g., MM_1, MM_2, ..., or MM_n). In this experiment, we set λ to 0.75, the number of malicious models C to 1, and M to 7. The number of nonmalicious models is $M - C$.

In the third experiment, the trust level of the selected models is studied as the number of models M is varied (5, 7, and 9) as illustrated in Fig. 15. In this experiment, we set λ to 0.75 and C to 1. In addition, the figure indicates that as M is increased, the trust level of the selected models is increased too.

Fig. 16 shows the results of our fourth experiment. In this experiment, the effect of using different values of λ (0.7, 0.75, and 0.8) on the trust level of the selected models is analyzed. In this experiment, we set C to 1 and M to 7. The figure shows this effect given different reconfiguration budgets B. The figure indicates that as λ is increased, the trust level of the selected models is increased too.

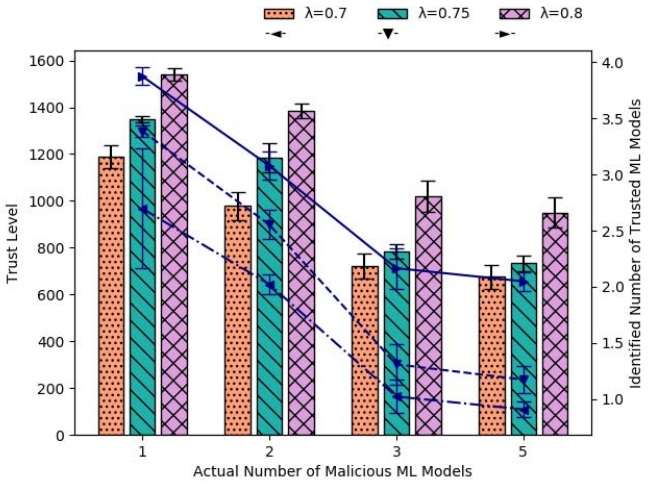

Fig. 17. IIoT PM use case: the effect of malicious models on the trust level.

Fig. 17 shows the effect of the number of malicious LSTM models C (1, 2, and 3) on the trust level of the selected models for different values of λ (0.7, 0.75, and 0.8). The figure also shows the actual number of malicious LSTM models versus the identified number of malicious LSTM models. In this experiment, we set M to 7 and B to 7.

C. Discussion and Lessons Learned

We can conclude the following based on the results presented in the previous section.

1) It is important to evaluate ML models used in critical and sensitive decisions in terms of trustworthiness and reliability. Additionally, other traditional criteria of the ML model evaluation must be considered (e.g., accuracy, run time, etc.).
2) Our proposed fixing heuristic strives to maximize the trust level while not affecting the accuracy of the selected models, as Figs. 8 and 13 indicate.
3) Figs. 9 and 14 show that our proposed fixing heuristic is able to obtain a trust level that is 0.7%–2.53% lower than that obtained by the upper bound solution in smart city case study, and 0.49%–3.17% lower than that obtained by the upper bound solution in the IIoT case study. Figs. 9 and 14 also indicate that by increasing the reconfiguration budget, the trust level is increased. However, there is a limit beyond which increasing the number of reconfigurations does not increase the trust level.
4) Figs. 10 and 15 indicate that increasing the number of selected models leads to an increase in the trust level of the overall system. This fact is similar to the concept of evaluating the seller feedback on online shopping sites, restaurants, or hotel reviews. As the volume of feedback increases, the level of reliability of such reviews increases as well.
5) Figs. 11 and 16 indicate that increasing λ, the number of the excluded models is decreased. However, increasing λ beyond a specific threshold may lead to the use of malicious models. On the other hand, using a small value for λ leads to excluding more models, which might not be malicious.
6) Figs. 12 and 17 indicate that increasing the number of malicious models leads to a decrease in the trust level of the overall system. This is due to the fact that the proposed heuristic excludes malicious models and it might reach a fail-safe execution state in which it informs the resource-constrained devices that there are no TML models to be hosted on them.

VII. Conclusion

In this article, we considered the paradigm in which resource-constrained IoT devices execute ML algorithms locally, without necessarily being connected to the cloud all the time. This paradigm is desirable in systems that have strict latency, connectivity, energy, privacy, and security requirements. There is a strong need in such environments to evaluate the level of trustworthiness of ML models built by different service providers, we formulated the problem of finding a subset of ML models that maximizes the trustworthiness while adhering to a given reconfiguration budget and rate constraints. We proved that this problem is NP-complete and propose a fixing heuristic that finds a near-optimal solution in polynomial time.

To measure the performance of the proposed fixing heuristic compared to ILP, we applied our proposed fixing heuristic to two different case studies: 1) the traffic flow volume data set to predict the number of vehicles (as a proxy case study for smart cities services) and 2) the turbofan engine degradation simulation data set to predict the remaining useful life for the engine (as a proxy for IIoT services). Our proposed fixing heuristic returns impressive performance achieving a high trust level that is less than the optimal ILP solution by only 0.7%–2.53% in the smart city service case study and 0.49%–3.17% less in the IIoT service case study.

There are a number of avenues of future work that can be pursued. Although we only use LSTM for developing the models in this article, other types of models (e.g., CNN, deep neural networks, and SVM) can also be explored. It would be interesting to perform a comparative study of these models and also consider their robustness to adversarial attacks compared to our proposed fixing heuristic. Additionally, potential applications of our proposed heuristic can be explored in the speech, video, and medical domains, and in recommendation systems.

References

[1] P. P. Ray, "A survey of IoT cloud platforms," *Future Comput. Informat. J.*, vol. 1, nos. 1–2, pp. 35–46, Dec. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2314728816300149
[2] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *Proc. 25th USENIX Conf. Security Symp.*, 2016, pp. 601–618. [Online]. Available: http://dl.acm.org/citation.cfm?id=3241094.3241142
[3] "Machine learning as a service market," Market Res. Engine, Deerfield Beach, FL, USA, Rep. TMMLSM1117, Nov. 2017. [Online]. Available: https://www.marketresearchengine.com/machine-learning-as-a-service-market

[4] G. Dán, R. B. Bobba, G. Gross, and R. H. Campbell, "Cloud computing for the power grid: From service composition to assured clouds," in *Proc. 5th USENIX Workshop Hot Topics Cloud Comput. (HotCloud)*, San Jose, CA, USA, Jun. 2013, pp. 1–6.

[5] R. I. Meneguette, "A vehicular cloud-based framework for the intelligent transport management of big cities," *Int. J. Distrib. Sens. Netw.*, vol. 12, no. 5, May 2016, Art. no. 8198597. [Online]. Available: https://doi.org/10.1155/2016/8198597

[6] J. Hanen, Z. Kechaou, and M. B. Ayed, "An enhanced healthcare system in mobile cloud computing environment," *Vietnam J. Comput. Sci.*, vol. 3, no. 4, pp. 267–277, Nov. 2016. [Online]. Available: https://doi.org/10.1007/s40595-016-0076-y

[7] X. Zhang *et al.,* "IEHouse: A non-intrusive household appliance state recognition system," in *Proc. IEEE SmartWorld Ubiquitous Intell. Comput. Adv. Trusted Comput. Scalable Comput. Commun. Cloud Big Data Comput. Internet People Smart City Innovat. (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*, San Francisco, CA, USA, Aug. 2017, pp. 1–8.

[8] D. Mourtzis, E. Vlachou, N. Milas, and N. Xanthopoulos, "A cloud-based approach for maintenance of machine tools and equipment based on shop-floor monitoring," *Procedia CIRP*, vol. 41, pp. 655–660, Jan. 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2212827115011488

[9] M. Barreno, B. Nelson, R. Sears, A. D. Joseph, and J. D. Tygar, "Can machine learning be secure," in *Proc. ACM Symp. Inf. Comput. Commun. Security (ASIACCS)*, Taipei, Taiwan, Mar. 2006, pp. 16–25.

[10] R. Lee, M. Assante, and T. Conway, *Analysis of the Cyber Attack on the Ukrainian Power Grid*, SANS Ind. Control Syst., Bethesda, MD, USA, Mar. 2016. [Online]. Available: https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[11] D. Kushner, *The Real Story of Stuxnet*, IEEE Spec. Technol. Eng. Sci. News, Piscataway, NJ, USA, Feb. 2013. [Online]. Available: https://spectrum.ieee.org/telecom/security/the-real-story-of-stuxnet

[12] *Seven Iranians Working for Islamic Revolutionary Guard Corps-Affiliated Entities Charged for Conducting Coordinated Campaign of Cyber Attacks Against U.S. Financial Sector*, U.S. Dept. Justice, Washington, DC, USA, Mar. 2016, [Online]. Available: https://www.justice.gov/opa/pr/seven-iranians-working-islamic-revolutionary-guard-corps-affiliated-entities-charged

[13] K. M. Khan and Q. Malluhi, "Establishing trust in cloud computing," *IT Prof.*, vol. 12, no. 5, pp. 20–27, Sep./Oct. 2010.

[14] S. Pearson, "Privacy, security and trust in cloud computing," in *Privacy and Security for Cloud Computing*, S. Pearson and G. Yee, Eds. London, U.K.: Springer, 2013, pp. 3–42.

[15] M. Chiregi and N. Jafari Navimipour, "Cloud computing and trust evaluation: A systematic literature review of the state-of-the-art mechanisms," *J. Elect. Syst. Inf. Technol.*, vol. 5, no. 3, pp. 608–622, Dec. 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2314717217300430

[16] J. Sidhu and S. Singh, "Compliance based trustworthiness calculation mechanism in cloud environment," *Procedia Comput. Sci.*, vol. 37, pp. 439–446, Jan. 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187705091401031X

[17] D. H. Mcknight and N. L. Chervany, "What is trust? a conceptual analysis and an interdisciplinary model," in *Proc. Amer. Conf. Inf. Syst.*, 2000, pp. 10–13.

[18] D. H. McKnight and N. L. Chervany, "What trust means in e-commerce customer relationships: An interdisciplinary conceptual typology," *Int. J. Electron. Commerce*, vol. 6, no. 2, pp. 35–59, Dec. 2001. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/10864415.2001.11044235

[19] P. Domingos, "A few useful things to know about machine learning," *Commun. ACM*, vol. 55, no. 10, p. 78, Oct. 2012. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2347736.2347755

[20] S. Kaul, "Speed and accuracy are not enough! trustworthy machine learning," in *Proc. AAAI/ACM Conf. AI Ethics Soc.*, New Orleans, LA, USA, Feb. 2018, pp. 372–373.

[21] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," Jul. 2016. [Online]. Available: http://arxiv.org/abs/1607.02533.

[22] G. F. Elsayed *et al.*, "Adversarial examples that fool both computer vision and time-limited humans," Feb. 2018. [Online]. Available: http://arxiv.org/abs/1802.08195.

[23] H. Hosseini, Y. Chen, S. Kannan, B. Zhang, and R. Poovendran, "Blocking transferability of adversarial examples in black-box learning systems," Mar. 2017. [Online]. Available: http://arxiv.org/abs/1703.04318.

[24] N. Moati, H. Otrok, A. Mourad, and J.-M. Robert, "Reputation-based cooperative detection model of selfish nodes in cluster-based QoS-OLSR protocol," *Wireless Pers. Commun.*, vol. 75, no. 3, pp. 1747–1768, Apr. 2014. [Online]. Available: http://link.springer.com/10.1007/s11277-013-1419-y

[25] *CityPulse Dataset Collection*. Accessed: Sep. 19, 2020. [Online]. Available: http://iot.ee.surrey.ac.uk:8080/

[26] A. Saxena and G. Goebel, C. C. Larrosa, and F.-K. Chang, *Turbofan Engine Degradation Simulation Data Set*, NASA Ames Res. Center, Mountain View, CA, USA, 2008. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

[27] B. Cline, R. S. Niculescu, D. Huffman, and B. Deckel, "Predictive maintenance applications for machine learning," in *Proc. Annu. Rel. Maintain. Symp. (RAMS)*, Orlando, FL, USA, Jan. 2017, pp. 1–7.

[28] R. Sipos, D. Fradkin, F. Moerchen, and Z. Wang, "Log-based predictive maintenance," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min. (KDD)*, 2014, pp. 1867–1876. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2623330.2623340

[29] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 865–873, Apr. 2015.

[30] A. Paul, N. Chilamkurti, A. Daniel, and S. Rho, "Chapter 8—Big data collision analysis framework," in *Intelligent Vehicular Networks and Communications*, A. Paul, N. Chilamkurti, A. Daniel, and S. Rho, Eds. Amsterdam, The Netherlands: Elsevier, Jan. 2017, pp. 177–184. [Online]. Available: http://www.sciencedirect.com/science/article/pii/B9780128092668000089

[31] N. Huq, R. Vosseler, and M. Swimmer, "Cyberattacks against intelligent transportation systems: Assessing future threats to ITS," TrendLabs, Shibuya City, Japan, Rep., 2017. Accessed: Sep. 19, 2020. [Online]. Available: https://documents.trendmicro.com/assets/white_papers/wp-cyberattacks-against-intelligent-transportation-systems.pdf

[32] T. Speicher, M. B. Zafar, K. P. Gummadi, A. Singla, and A. Weller, "Reliable learning by subsuming a trusted model: Safe exploration of the space of complex models," in *Proc. Int. Conf. Mach. Learn. Workshop (ICML)*, Sydney, NSW, Australia, Aug. 2017, pp. 1–5.

[33] S. Ghosh, P. Lincoln, A. Tiwari, and X. Zhu, "Trusted machine learning for probabilistic models," in *Proc. Rel. Mach. Learn. Wild ICML*, New York, NY, USA, Jun. 2016, pp. 1–5. [Online]. Available: https://sites.google.com/site/wildml2016/ghosh16trusted.pdf

[34] X. Zhang, X. Zhu, and S. J. Wright, "Training set debugging using trusted items," Jan. 2018. [Online]. Available: https://arxiv.org/abs/1801.08019.

[35] M. T. Ribeiro, S. Singh, and C. Guestrin, "'Why should i trust you?': Explaining the predictions of any classifier," Feb. 2016. [Online]. Available: http://arxiv.org/abs/1602.04938.

[36] U. Jayasinghe, G. M. Lee, T.-W. Um, and Q. Shi, "Machine learning based trust computational model for IoT services," *IEEE Trans. Sustain. Comput.*, vol. 4, no. 1, pp. 39–52, Jan.–Mar. 2019.

[37] A. Fariha, A. Tiwari, A. Radhakrishna, S. Gulwani, and A. Meliou, "Data invariants: On trust in data-driven systems," Mar. 2020. [Online]. Available: http://arxiv.org/abs/2003.01289.

[38] J. Drozdal *et al.*, "Trust in AutoML: Exploring information needs for establishing trust in automated machine learning systems," in *Proc. 25th Int. Conf. Intell. User Interfaces*, Cagliari, Italy, Mar. 2020, pp. 297–307. [Online]. Available: http://arxiv.org/abs/2001.06509

[39] O. A. Wahab, J. Bentahar, H. Otrok, and A. Mourad, "Optimal load distribution for the detection of VM-based DDoS attacks in the cloud," *IEEE Trans. Services Comput.*, vol. 13, no. 1, pp. 114–129, Jan./Feb. 2020.

[40] T. Liu *et al.*, "Vision-based semi-supervised homecare with spatial constraint," in *Advances in Multimedia Information Processing—PCM 2008* (Lecture Notes in Computer Science), Y.-M. R. Huang *et al.*, Eds. Heidelberg, Germany: Springer, 2008, pp. 416–425.

[41] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," Sep. 2019. [Online]. Available: http://arxiv.org/abs/1706.06083.

[42] S. G. Finlayson, J. D. Bowers, J. Ito, J. L. Zittrain, A. L. Beam, and I. S. Kohane, "Adversarial attacks on medical machine learning," *Science*, vol. 363, no. 6433, pp. 1287–1289, Mar. 2019, [Online]. Available: https://science.sciencemag.org/content/363/6433/1287

[43] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Security Artif. Intell.*, Oct. 2011, pp. 43–58. [Online]. Available: https://doi.org/10.1145/2046684.2046692

[44] P. Saadatpanah, A. Shafahi, and T. Goldstein, "Adversarial attacks on copyright detection systems," Jun. 2019. [Online]. Available: http://arxiv.org/abs/1906.07153.

[45] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, Mar. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S209580991930503X

[46] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," Sep. 2018. [Online]. Available: http://arxiv.org/abs/1810.00069.

[47] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," Jan. 2018. [Online]. Available: http://arxiv.org/abs/1801.00553.

[48] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.

[49] M. R. Forster, "Key concepts in model selection: Performance and generalizability," *J. Math. Psychol.*, vol. 44, no. 1, pp. 205–231, Mar. 2000. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0022249699912841

[50] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," in *Data Mining and Constraint Programming: Foundations of a Cross-Disciplinary Approach* (Lecture Notes in Computer Science), C. Bessiere, L. De Raedt, L. Kotthoff, S. Nijssen, B. O'Sullivan, and D. Pedreschi, Eds. Cham, Switzerland: Springer, 2016, pp. 149–190.

[51] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Dis. Data Min. (KDD)*, Chicago, IL, USA, Aug. 2013, pp. 847–855. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2487575.2487629

[52] B. Komer, J. Bergstra, and C. Eliasmith, "Hyperopt-sklearn: Automatic hyperparameter configuration for scikit-learn," in *Proc. 13th Python Sci. Conf. (Scipy)*, Austin, TX, USA, Jul. 2014, pp. 32–37.

[53] E. R. Sparks *et al.*, "MLI: An API for distributed machine learning," in *Proc. IEEE 13th Int. Conf. Data Min.*. Dallas, TX, USA, Dec. 2013, pp. 1187–1192. [Online]. Available: http://ieeexplore.ieee.org/document/6729619/

[54] P. Lokuciejewski, M. Stolpe, K. Morik, and P. Marwedel, "Automatic selection of machine learning models for compiler heuristic generation," in *Proc. 4th Workshop Stat. Mach. Learn. Approaches Archit. Compilation (SMART)*, Pisa, Italy, Jan. 2010, pp. 3–17.

[55] R. Leite, P. Brazdil, and J. Vanschoren, "Selecting classification algorithms with active testing," in *Machine Learning and Data Mining in Pattern Recognition* (Lecture Notes in Computer Science), P. Perner, Ed. Heidelberg, Germany: Springer, 2012, pp. 117–131.

[56] J. N. van Rijn, S. M. Abdulrahman, P. Brazdil, and J. Vanschoren, "Fast algorithm selection using learning curves," in *Advances in Intelligent Data Analysis XIV* (Lecture Notes in Computer Science), E. Fromont, T. De Bie, and M. van Leeuwen, Eds. Cham, Switzerland: Springer, 2015, pp. 298–309.

[57] A. Kumar, S. Goyal, and M. Varma, "Resource-efficient machine learning in 2 KB RAM for the Internet of Things," in *Proc. 34th Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Jul. 2017, pp. 1935–1944. [Online]. Available: http://proceedings.mlr.press/v70/kumar17a.html

[58] C. Gupta *et al.*, "ProtoNN: Compressed and accurate kNN for resource-scarce devices," in *Proc. Int. Conf. Mach. Learn.*, Sydney, NSW, Australia, Jul. 2017, pp. 1331–1340. [Online]. Available: http://proceedings.mlr.press/v70/gupta17a.html

[59] M. Motamedi, D. Fong, and S. Ghiasi, "Machine intelligence on resource-constrained IoT devices: The case of thread granularity optimization for CNN inference," *ACM Trans. Embedded Comput. Syst.*, vol. 16, no. 5s, pp. 1–19, Sep. 2017. [Online]. Available: http://doi.acm.org/10.1145/3126555

[60] W. Meng, Z. Gu, M. Zhang, and Z. Wu, "Two-bit networks for deep learning on resource-constrained embedded devices," Jan. 2017. [Online]. Available: https://arxiv.org/abs/1701.00485.

[61] A. Shoeb and J. Guttag, "Application of machine learning to epileptic seizure detection," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 975–982. [Online]. Available: http://dl.acm.org/citation.cfm?id=3104322.3104446

[62] B. Li, R. Lu, W. Wang, and K.-K. R. Choo, "Distributed host-based collaborative detection for false data injection attacks in smart grid cyber-physical system," *J. Parallel Distrib. Comput.*, vol. 103, pp. 32–41, May 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743731516301885

[63] J.-H. Cho, I.-R. Chen, and K. S. Chan, "Trust threshold based public key management in mobile ad hoc networks," *Ad Hoc Netw.*, vol. 44, pp. 58–75, Jul. 2016. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1570870516300555

[64] M. Raya, P. Papadimitratos, V. D. Gligor, and J. Hubaux, "On data-centric trust establishment in ephemeral ad hoc networks," in *Proc. IEEE INFOCOM 27th Conf. Comput. Commun.*, Phoenix, AZ, USA, Apr. 2008, pp. 1238–1246.

[65] A. Srinivasan, J. Teitelbaum, and J. Wu, "DRBTS: Distributed reputation-based beacon trust system," in *Proc. 2nd IEEE Int. Symp. Depend. Auton. Secure Comput.*, Indianapolis, IN, USA, Sep. 2006, pp. 277–283.

[66] R. K. Shahzad and N. Lavesson, "Comparativeanalysis of voting schemes for ensemble-based malware detection," *J. Wireless Mobile Netw. Ubiquitous Comput. Depend. Appl.*, vol. 4, pp. 98–117, Mar. 2013.

[67] B. Qolomany, M. Maabreh, A. Al-Fuqaha, A. Gupta, and D. Benhaddou, "Parameters optimization of deep learning models using particle swarm optimization," in *Proc. 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Valencia, Spain, Jun. 2017, pp. 1285–1290.

[68] B. Qolomany, A. Al-Fuqaha, D. Benhaddou, and A. Gupta, "Role of deep LSTM neural networks and Wi-Fi networks in support of occupancy prediction in smart buildings," in *Proc. IEEE 19th Int. Conf. High Perform. Comput. Commun. 15th Int. Conf. Smart City 3rd Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Bangkok, Thailand, Dec. 2017, pp. 50–57.

[69] B. Qolomany *et al.*, "Leveraging machine learning and big data for smart buildings: A comprehensive survey," *IEEE Access*, vol. 7, pp. 90316–90356, 2019.

[70] T. Pyzdek, *The Six Sigma Handbook: The Complete Guide for Greenbelts, Blackbelts, and Managers at All Levels. Revised and Expanded Edition*, 2nd ed. New York, NY, USA: McGraw-Hill, Mar. 2003.

Basheer Qolomany (Member, IEEE) received the B.Sc. and M.Sc. degrees in computer science from the University of Mosul, Mosul, Iraq, in 2008 and 2011, respectively, and the Ph.D. and second master's en-route to Ph.D. degrees in computer science from Western Michigan University (WMU), Kalamazoo, MI, USA, in 2018.

He served as a Visiting Assistant Professor with the Department of Computer Science, Kennesaw State University, Marietta, GA, USA, from 2018 to 2019, and a Graduate Doctoral Assistant with the Department of Computer Science, WMU from 2016 to 2018. He also served as a Lecturer with the Department of Computer Science, University of Duhok, Kurdistan, Iraq, from 2011 to 2013. He is currently an Assistant Professor with the Department of Cyber Systems, University of Nebraska at Kearney, Kearney, NE, USA. His research interests include machine learning, deep learning, Internet of Things, smart services, cloud computing, and big data analytics.

Dr. Qolomany has served as a reviewer of multiple journals, including the IEEE INTERNET OF THINGS JOURNAL, *Energies*, *Open Access Journal*, and *Computers and Electrical Engineering Journal* (Elsevier). He also served as a Technical Program Committee member and a reviewer of some international conferences, including IEEE Globecom, IEEE IWCMC, and IEEE VTC.

Ihab Mohammed (Member, IEEE) received the B.S. and M.S. degrees in computer science from Al-Nahrain University, Baghdad, Iraq, in 2002 and 2005, respectively. He is currently pursuing the Ph.D. degree with the NEST Research Laboratory, Computer Science Department, Western Michigan University, Kalamazoo, MI, USA.

His current research interests include the design, simulation, and analysis of algorithms in the fields of computer networks, Internet of Things, vehicular networks, and big data.

Ala Al-Fuqaha (Senior Member, IEEE) received the Ph.D. degree in computer engineering and networking from the University of Missouri–Kansas City, Kansas, MO, USA, in 2004.

He is currently a Professor with Hamad Bin Khalifa University, Doha, Qatar, and Western Michigan University, Kalamazoo, MI, USA. His research interests include the use of machine learning in general and deep learning in particular in support of the data-driven and self-driven management of large-scale deployments of IoT and smart city infrastructure and services, wireless vehicular networks (VANETs), cooperation and spectrum access etiquette in cognitive radio networks, and management and planning of software defined networks.

Prof. Al-Fuqaha is an ABET Program Evaluator. He serves on editorial boards of multiple journals, including IEEE COMMUNICATIONS LETTERS and *IEEE Network Magazine*. He also served as a chair, a co-chair, and a technical program committee member of multiple international conferences, including IEEE VTC, IEEE Globecom, IEEE ICC, and IWCMC.

Mohsen Guizani (Fellow, IEEE) received the B.S. and M.S. degrees in electrical engineering and the M.S. and Ph.D. degrees in computer engineering from Syracuse University, Syracuse, NY, USA, in 1984, 1986, 1987, and 1990, respectively.

He was the Associate Vice President of Qatar University, Doha, Qatar; the Chair of the Computer Science Department, Western Michigan University, Kalamazoo, MI, USA; and the Computer Science Department, University of West Florida, Pensacola, FL, USA, and the Director of Graduate Studies with the University of Missouri–Columbia, Columbia, MO, USA. He is currently a Professor with the Department of Computer Science and Engineering, Qatar University. He has authored or coauthored nine books and publications in refereed journals and conferences. His research interests include wireless communications and mobile computing, vehicular communications, smart grid, cloud computing, and security.

Junaid Qadir (Senior Member, IEEE) received the bachelor's degree in electrical engineering from UET, Lahore, Pakistan, in 2000, and the Ph.D. degree from the University of New South Wales, Sydney, NSW, Australia, in 2008.

He is a Professor with the Information Technology University, Lahore, Pakistan, where he is the Director of IHSAN (ICTD; Human Development; Systems; Big Data Analytics; Networks Lab) Research Lab. His primary research interests are in the areas of computer systems and networking and using ICT for development.

Prof. Qadir is an Award-Winning Teacher who has been Awarded the Highest National Teaching Award in Pakistan—the Higher Education Commission's Best University Teacher Award—for the year 2012–2013. He has considerable teaching experience and a wide portfolio of taught courses in the disciplines of systems and networking, signal processing, and wireless communications and networking. He is an Associate Editor of IEEE ACCESS, *Nature Central's Big Data Analytics* (Springer), *Human-Centric Computing and Information Sciences* (Springer), and *IEEE Communications Magazine*. He has been appointed as an ACM Distinguished Speaker from 2020 to 2022. He has served on the program committee of a number of international conferences and reviews regularly for various high-quality journals. He is a Senior Member of ACM.