Contextual-Bandit Anomaly Detection for IoT Data in Distributed Hierarchical Edge Computing

Mao V. Ngo*†, Tie Luo‡, Hakima Chaouchi§, and Tony Q.S. Quek*
*Singapore University of Technology and Design, Singapore
†Institute for Infocomm Research, A*STAR, Singapore
‡Department of Computer Science, Missouri University of Science and Technology, USA
§CNRS, SAMOVAR, Telecom Sud Paris, Institut Mines Telecom, Paris-Saclay University, France
vanmao_ngo@mymail.sutd.edu.sg, tluo@mst.edu,
hakima.chaouchi@telecom-sudparis.eu, tonyquek@sutd.edu.sg

***Abstract*—Advances in deep neural networks (DNN) greatly bolster real-time detection of anomalous IoT data. However, IoT devices can hardly afford complex DNN models, and offloading anomaly detection tasks to the cloud incurs long delay. In this paper, we propose and build a demo for an adaptive anomaly detection approach for distributed hierarchical edge computing (HEC) systems to solve this problem, for both univariate and multivariate IoT data. First, we construct multiple anomaly detection DNN models with increasing complexity, and associate each model with a layer in HEC from bottom to top. Then, we design an adaptive scheme to select one of these models on the fly, based on the contextual information extracted from each input data. The model selection is formulated as a *contextual bandit problem* characterized by a single-step Markov decision process, and is solved using a *reinforcement learning policy network*. We build an HEC testbed, implement our proposed approach, and evaluate it using real IoT datasets. The demo shows that our proposed approach significantly reduces detection delay (e.g., by 71.4% for univariate dataset) without sacrificing accuracy, as compared to offloading detection tasks to the cloud. We also compare it with other baseline schemes and demonstrate that it achieves the best accuracy-delay tradeoff.** [1]

I. Introduction

With the increasing demand of detecting anomalous sensory data generated by a massive number of IoT devices, machine learning—especially deep learning—offers an effective approach and has been successfully applied to many anomaly detection tasks in IoT environments [1]–[3]. A variety of IoT applications, such as collision avoidance for autonomous vehicles and fire alarm systems in factories, are time-critical and require fast anomaly detection. In these cases, the traditional approach of streaming all the IoT sensory data to the cloud can be problematic as it tends to incur high communication delay, congest backbone networks, and pose data privacy threats.

Anomaly detection (AD) with edge or fog computing [4], [5] provides an alternative by performing distributed AD in the proximity of sensory data sources. However, pushing computation from cloud to the edge faces resource challenges especially when complex deep learning models are used. Remedies include model compression [6] or successive offloading in a hierarchy until a certain performance threshold is reached [7]. But overall, there are three main issues in existing works: (1) "one size fits all" - attempting to use one AD model to handle all the input data, while overlooking the fact that different data samples often have different levels of hardness in detecting anomaly events; (2) focusing on accuracy or F1-score without giving adequate consideration on detection delay and memory footprint; (3) lacking appropriate local analysis and thus often transmitting data back and forth between sources and the cloud, incurring unnecessary delay and bandwidth consumption.

In this paper, we propose an adaptive distributed approach that leverages the hierarchical edge computing (HEC) architecture by adaptively matching data of different hardness levels of detection with models of different complexity. Specifically, we construct multiple anomaly detection DNN models (using autoencoder and LSTM) of increasing complexity and associate them with the multiple layers of HEC from bottom to top, i.e., IoT devices, edge servers, and the cloud. Then, we propose an adaptive scheme that judiciously selects one of these models to perform AD at the most suitable layer, based on the contextual information extracted from each input data on the fly. The scheme follows a single-step Markov decision process (hence can make quick decisions) derived from a *contextual bandit problem* that we formulate and solve using a *reinforcement learning policy network*. By selecting appropriate models, we avoid unnecessary data transmissions between IoT devices, edge servers, and the cloud, while maintaining the best detection accuracy. We build an HEC testbed and implement our proposed approach for both univariate and multivariate data.[1] Our extensive evaluation using real-world IoT datasets demonstrate that our proposed approach achieves the best tradeoff between high detection accuracy and low detection delay, and outperforms multiple other benchmark schemes.

[1]Our demo is also available online: https://rebrand.ly/91a71

II. Design

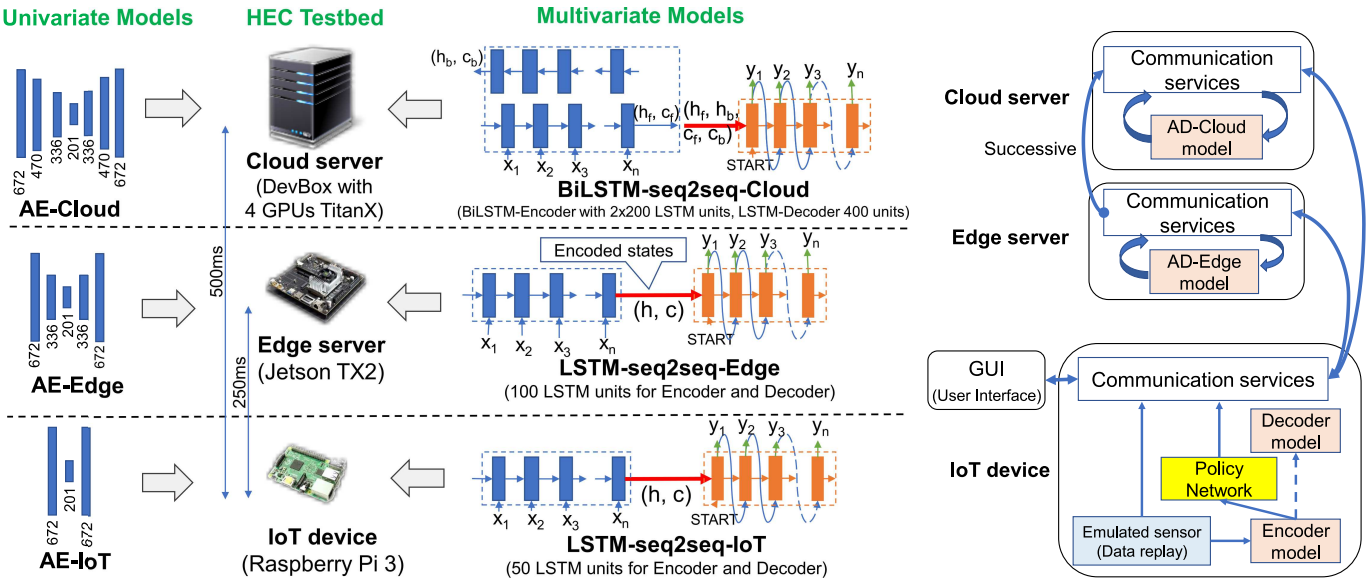
We consider a K-layer distributed hierarchical edge computing (HEC), and choose $K = 3$ as a typical setting [4] (but our approach applies to any K in general, i.e., multiple layers of edge servers). We build an HEC testbed with three layers consisting of a Raspberry Pi 3 as the IoT device, an NVIDIA Jetson-TX2 as the edge server, and an NVIDIA Devbox (with 4 GPU TitanX) as the cloud server, as shown in Fig. 1a.

We consider two types of data: univariate and multivariate. We construct $K = 3$ AD models for each type, with increasing complexity, and associate those models with the HEC layers from 1 to K. Then, we design an adaptive model selection scheme to choose the most suitable model on the fly to detect anomalies.

A. Constructing Anomaly Detection Models in HEC

1) AD Models for Univariate Data: We use autoencoders (AE) to build AD models for univariate IoT data, as the feasibility of running such models on IoT devices has been proved by [1]. Hence, we build three AE-based models called *AE-IoT*, *AE-Edge*, and *AE-Cloud*, respectively, to associate with the three corresponding layers of HEC. These models have three, five, seven layers and thus have different capabilities of learning features for data representation. See Fig. 1a.

2) AD Models for Multivariate Data: The simplicity of AE-based models does not well represent features for high-dimensional IoT data. Hence in the multivariate case (18 dimensions in our demo), we use sequence-to-sequence (seq2seq) model [8] based on long short-term memory (LSTM) to build an LSTM encoder-decoder as the AD model. Such models can learn representations of multi-sensor time-series data with 1 to 12 dimensions [2]. In our case, we apply our LSTM-seq2seq model to an 18-dimensional dataset, and deploy the model on the IoT device, and name it *LSTM-seq2seq-IoT*. The multivariate IoT data are encoded into encoded states by an LSTM-encoder, then an LSTM-decoder learns to reconstruct data from previous encoded states and previous output, one step a time (for the first step, a special token is used, which in our case is a



(a) Testbed hierarchical edge computing (HEC) and architecture of anomaly detection (AD) models

(b) Software architecture of the demo.

Fig. 1: The HEC testbed and architecture of AD models for univariate and multivariate data. Demo software architecture.

zero vector). For the edge layer, we build an *LSTM-seq2seq-Edge* anomaly detection model with double number of LSTM units for both encoder and decoder, which can learn a better representation of a longer sequence input. For the cloud layer, we build a *BiLSTM-seq2seq-Cloud* anomaly detection model with a bidirectional-LSTM (BiLSTM) encoder to learn both backward and forward directions of the input sequence to encode information into encoded states. These are depicted in Fig. 1a. To train these LSTM-seq2seq models, we use the `RMSProp` optimizer and ℓ_2-norm kernel regularizer of $1e-4$ to minimize the mean squared reconstruction error. The output of LSTM-decoder is dropped out with a drop-rate 0.3, and then passes through a fully-connected layer with the linear activation function to generate a reconstruction sequence.

3) Anomaly Score: We assume that reconstruction errors follow the Gaussian distribution $\mathcal{N}(\mu, \mathbf{\Sigma})$, where μ and $\mathbf{\Sigma}$ are the mean and covariance matrix of reconstruction errors of normal data samples. We use *logarithmic probability densities (logPD)* of the reconstruction errors as *anomaly scores*, as is similar to [2], [3], [9]. We then use the minimum value of the logPD on the normal dataset (i.e., the training set) as the threshold for detecting outliers.

We consider a detection as confident if the input sequence being detected satisfies one of these two conditions: (i) at least one data point has a logPD of less than certain times (e.g., 2x) of the threshold (note logPD is negative); (ii) the number of anomalous points is higher than a certain percentage (e.g., 5%) of the sequence size.

B. Adaptive Model Selection Scheme

We propose an adaptive model selection scheme to select the most suitable AD model based on the contextual information of input data, so that each data sample will be directly fed to its best-suited model. This is in contrast to traditional approaches where input data either (i) always go to one same model regardless of the hardness of detection [5], or (ii) are successively offloaded to higher layers until meeting a desired accuracy or confidence [7].

Our proposed adaptive scheme is a reinforcement learning algorithm that adapts its model selection strategy to maximize the expected reward of a model to be selected. We frame the learning problem as a *contextual bandit problem* [10], [11] and use a *policy gradient method* to solve it. See Fig. 2. Formally, given the contextual information $\mathbf{z_x}$ of an input data $\mathbf{x}$, and K trained AD models deployed at the K layers of an HEC system, we build a policy network that takes $\mathbf{z_x}$ as the input state and outputs a policy of selecting which model (or equivalently which layer of HEC) to do anomaly detection, in the form of a categorical distribution $\pi_\theta(\mathbf{a}|\mathbf{z_x}) = \prod_{k=1}^{K} s_k^{a_k}$, where $\mathbf{a}$ is the actions encoded as a one-hot vector that defines which model (or HEC layer) to perform the task, $\mathbf{s} = (s_1, s_2, \cdots, s_K) = f_\theta(\mathbf{z_x})$, is a vector representing the likelihood of selecting each model k. We denote the selected action as $|\mathbf{a}| = k$ if $k = \arg\max_k(s_k)$.

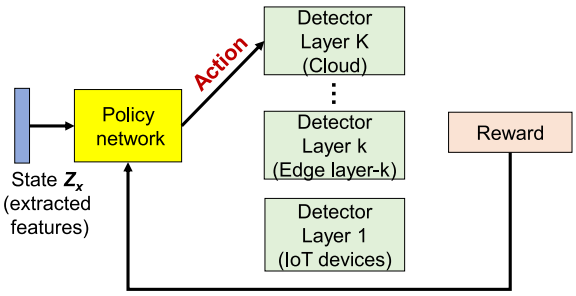


Fig. 2: Adaptive model selection with a policy network.

The policy network $f_\theta(.)$ is designed as a neural networks with parameters θ. To make the policy network small enough to run fast on IoT devices, we use the extracted features $\mathbf{z_x}$ instead of the raw input data $\mathbf{x}$, to represent the contextual information of input data. We train the policy network using policy gradient method [10], [11] to find an optimal policy π that maps an input state $\mathbf{z_x}$ to an action (which model or layer) to minimize the negative expected reward:

$$\min L(\theta) = - \mathop{\mathbb{E}}_{\mathbf{a}\sim\pi_\theta} [R(\mathbf{a}, \mathbf{z_x})],$$

where $R(\mathbf{a}, \mathbf{z_x})$ is a reward function of action $\mathbf{a}$ given state $\mathbf{z_x}$. To reduce the variance of reward value and increase the convergence rate, we utilize *reinforcement comparison* [11] with a baseline $R(\tilde{\mathbf{a}}, \mathbf{z_x})$. In order to encourage selecting an appropriate model that jointly increases accuracy and reduces the cost of offloading tasks further away from the edge, we propose a reward function as follows: $R(\mathbf{a}, \mathbf{z_x}) = \text{accuracy}(\mathbf{x}) - C(\mathbf{a}, \mathbf{x})$. We define the cost function $C(\mathbf{a}, \mathbf{x})$ as a function maps the end-to-end detection delay $t_{e2e}(\mathbf{x}, \mathbf{a})$

to an equivalent accuracy in scale $[0, 1]$ with intuition that a higher delay will result in a greater reduction of accuracy:

$$C(\mathbf{a}, \mathbf{x}) = \frac{\alpha \cdot t_{\text{e2e}}(\mathbf{x}, \mathbf{a})}{1 + \alpha \cdot t_{\text{e2e}}(\mathbf{x}, \mathbf{a})}, \quad (1)$$

where α is a tunable parameter. Further details are available in [3].

III. Implementation and Experiment Setup

A. Dataset

We evaluate our proposed approach with two public datasets. The data is standardized to zero mean and unit variance for all of the above training tasks and datasets.

Univariate dataset. We use a dataset on power consumption,[2] which is used in [2], [3], [9]. For training details, please refer to our previous work [3].

Multivariate dataset. We use MHEALTH[3] which consists of 12 human activities of 10 different people, and each person wore two motion sensors: one on left-ankle and the other on right-wrist. Each motion sensor contains a 3-axis accelerator, a 3-axis gyroscope, and a 3-axis magnetometer; hence the input data has 18 dimensions. The sampling rate of these sensors is 50 Hz. We use a window sequence of 128 time-steps (∼2.56 second) with a step-size of 64. Adopting the common practice, we choose the dominant human activity (e.g., walking) as normal and treat the other activities as anomalous. For the AD task, we select 70% of normal samples of all the subjects (people) as the training set; and the rest 30% of normal samples plus 5% of each of the other activities as the test set. To train the policy network, we select 30% of normal samples and 5% of each of the other activities as the training set, and the whole dataset as the test set.

B. Implementation

We use Tensorflow and Keras to implement the AD models (i.e., three AE models and three LSTM-seq2seq models as shown in Fig. 1a) and the policy network model. Note that, since the edge server and cloud server are empowered with GPU, we implement the LSTM-seq2seq-Edge and BiLSTM-seq2seq-Cloud models based on CuDNNLSTM units to accelerate training and inference time. Before deploying the LSTM-seq2seq-IoT and LSTM-seq2seq-Edge models on Raspberry Pi3 and Jetson-TX2, we compress them by (i) removing the trainable nodes from the graph, and convert variables into constants; (ii) quantizing the model parameters from floating-point 32-bit (FP32) to FP16. We observe no performance decrease of these compressed AD models running on Raspberry Pi3 and Jetson-TX2.

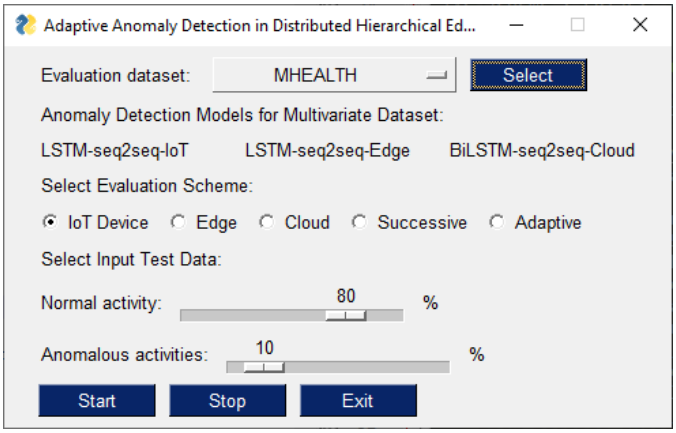
The policy network requires low complexity and needs to run fast on IoT devices, so the state input to the policy network needs to be small but still well represent the whole sequence of input data. For the univariate data, we define the contextual state as an extracted feature vector which includes min, max, mean, and standard deviation of each day's sensor data. For the multivariate data, we use the encoded states of the LSTM-encoder to represent the input state for the policy network. Subsequently, we build the policy network as a single hidden neural network with 100 hidden units and an output layer with 3 units. We empirically select $\alpha = 0.0005$ for the univariate dataset, and $\alpha = 0.00035$ for the multivariate dataset to calculate the cost of executing detection as given by (1).

C. Software Architecture and Experiment Setup

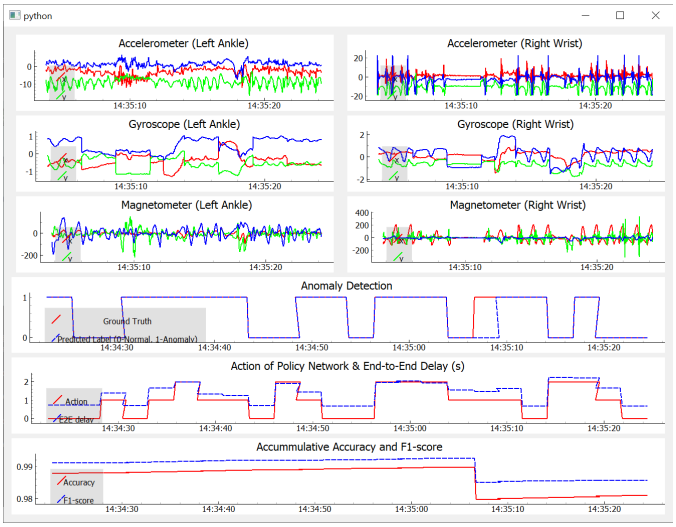
The software architecture of our demo is shown in Fig. 1b. It consists of a GUI, the adaptive model selection scheme based on the policy network, and the three AD models at the three layers of HEC. The GUI allows a user to select which dataset and model selection

[2]http://www.cs.ucr.edu/~eamonn/discords/

[3]http://archive.ics.uci.edu/ml/datasets/mhealth+dataset



(a) GUI: user can select dataset and evaluation scheme, and tune parameters.



(b) Result panel: raw sensory signals, AD performance and associated actions.

Fig. 3: Demo: GUI and results: multivariate data.

scheme to use, as well as to tune parameters, as shown in Fig. 3a, and displays the sensory raw signals and performance results, as shown in Fig. 3b. All the communication services use keep-alive TCP sockets to reduce the overhead of connection establishment. Network latency as shown in Fig. 1a is configured by using Linux traffic control tool, `tc`, to emulate the WAN connections in HEC. The hardware setup for the HEC testbed is shown in Fig. 4.

User Actions: As shown in Fig. 3a, we allow users (e.g., ICDCS demo session participants) to evaluate the HEC testbed performance with (i) either univariate or multivariate datasets, (ii) different fractions of normal and abnormal data in the datasets to use, and (iii) different schemes under evaluation: (1) always detects anomaly at *IoT Device*, (2) always offloads detection tasks to *Edge* server, (3) always offloads to *Cloud*, (4) *Successive*, i.e., executes at IoT devices first and then offloads to higher layers successively until reaching a confident output or the cloud, and (5) *Adaptive* which is our proposed adaptive model selection scheme. After the user clicks "Start", our result panel as in Fig. 3b will show the continuously updated raw sensory data (accelerometer, gyroscope, magnetometer), anomaly detection outcome (0 or 1) vs. ground truth, detection delay vs. the actions determined by our policy network, and the accumulative accuracy and F1-score.

IV. Experiment Results

Comparison among AD models: Table I compares the performance and complexity among the three models we use. For

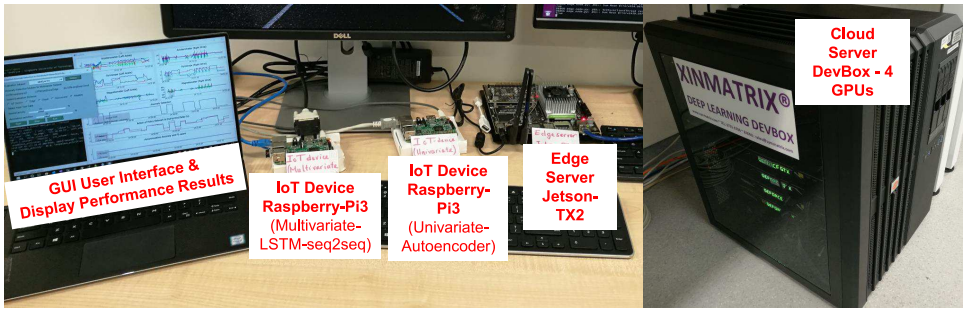


Fig. 4: Hardware setup of our HEC testbed.

TABLE I: Comparison among AD models.

Dataset/Model	Univariate/Autoencoder			Multivariate/LSTM-seq2seq		
Layer	**IoT**	**Edge**	**Cloud**	**IoT**	**Edge**	**Cloud**
#Parameters	271,017	949,468	1,085,077	28,518	97,818	1,028,018
Accuracy(%)	78.09	93.33	98.09	82.63	94.21	97.37
F1-score	0.465	0.741	0.909	0.852	0.955	0.980
Exec time (ms)	12.4	7.4	4.5	591.0	417.3	232.3

both univariate and multivariate data, the complexity of AD models increases from IoT to cloud, as indicated by "# of Parameters" (weights and biases) which reflects the approximate *memory footprint* of the models. Along with this, the F1-score and accuracy increase as well; for example, AE-Cloud are 95.5% and 20% higher than AE-IoT, and BiLSTM-seq2seq-Cloud are 15% and 14.74% higher than LSTM-seq2seq-IoT on these two metrics. On the other hand, the execution time (for running the detection algorithms) decreases from IoT to cloud, as indicated in the last row of Table I, which is measured on the actual machines of our HEC testbed and averaged over five runs. This is due to the different computation capacity (whereas communication capacity is taken into account by our end-to-end delay shown later). One more observation is that LSTM-seq2seq models, which handle multivariate datasets, take much longer time to run (up to 591 ms) than AE models, which handle univariate datasets (up to 12.4 ms).

Comparison among model selection schemes: We can see in Table II that the IoT Device scheme achieves the lowest detection delay but also the poorest accuracy and F1-score among all the evaluated schemes. On the other extreme, the Cloud scheme yields the best accuracy and F1-score but incurs the highest detection delay (end-to-end). The Successive scheme leverages distributed anomaly detectors in HEC and thus significantly reduces the average detection delay as compared to the Edge and Cloud schemes. However, its accuracy and F1-score are outperformed by the Edge scheme. In contrast, our proposed adaptive scheme not only achieves lower detection delay but its F1-score and accuracy also consistently outperform those of IoT Device, Edge, and Successive schemes. Even though the F1-score and accuracy of our proposed scheme are marginally lower than those of the Cloud scheme (e.g., 0.82% and 0.4% for the multivariate dataset), our scheme reduces the end-to-end detection delay by a substantial 71.4% and 7.84% for the univariate and multivariate datasets, respectively. In summary, and as also indicated in the last column "Reward", which is a convex combination of both accuracy and delay, our proposed adaptive scheme strikes the best tradeoff between accuracy and detection delay.

V. Conclusion

In this paper, we identify three issues in existing IoT anomaly detection (AD) approaches, and propose an adaptive AD approach for IoT data in HEC for both univariate and multivariate datasets. We construct multiple distributed AD models based on autoencoder and LSTM with increasing complexity, and associate them with the HEC layers from bottom to top. Next, it uses a reinforcement learning based adaptive scheme to select the best-suited model based on the contextual information of input data. The scheme consists of a policy network as the solution to a contextual bandit problem, characterized by a single-step MDP. We build an HEC testbed and conduct our demo experiments using two real-world IoT datasets. By comparing with other baseline schemes, we show that our proposed scheme strikes the best performance tradeoff between detection accuracy and detection delay, which is a typical dilemma.

TABLE II: Comparison among AD model detection schemes.

Dataset	Scheme	F1	Accuracy(%)	Delay(ms)	Reward
Univariate	**IoT Device**	0.465	93.68	12.4	48.39
	Edge	0.800	98.63	257.43	45.36
	Cloud	0.909	99.46	504.50	41.24
	Successive	0.769	98.35	105.27	N/A
	Our Method	0.870	99.17	144.50	**49.52**
Multivariate	**IoT Device**	0.848	93.19	591.0	389.85
	Edge	0.951	97.59	667.30	403.77
	Cloud	0.980	99.00	732.30	404.12
	Successive	0.911	95.79	626.16	N/A
	Our Method	0.972	98.60	674.87	**408.06**

References

[1] T. Luo and S. G. Nagarajan, "Distributed anomaly detection using autoencoder neural networks in WSN for IoT," in *IEEE ICC*, May 2018.

[2] P. Malhotra *et al.*, "LSTM-based encoder-decoder for multi-sensor anomaly detection," *ICML Workshop*, 2016.

[3] M. V. Ngo, H. Chaouchi, T. Luo, and T. Q. S. Quek, "Adaptive anomaly detection for IoT data in hierarchical edge computing," in *AAAI Workshop*, New York, NY, 2020.

[4] Q. D. La, M. V. Ngo, T. Q. Dinh, T. Q. Quek, and H. Shin, "Enabling intelligence in fog computing to achieve energy and latency reduction," *Digital Communications and Networks*, vol. 5, no. 1, pp. 3–9, 2019.

[5] Z. Chen *et al.*, "An empirical study of latency in an emerging class of edge computing applications for wearable cognitive assistance," in *Proc. ACM/IEEE SEC*, 2017.

[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *ICLR*, 2016.

[7] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE ICDCS*, 2017, pp. 328–339.

[8] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014.

[9] A. Singh, "Anomaly detection for temporal data using long short-term memory (LSTM)," Master's thesis, 2017.

[10] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. NIPS*, 2000, pp. 1057–1063.

[11] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.