Federated Multi-Discriminator BiWGAN-GP based Collaborative Anomaly Detection for Virtualized Network Slicing

Weili Wang, Chengchao Liang, Lun Tang, Halim Yanikomeroglu, *Fellow, IEEE*, Qianbin Chen, *Senior Member, IEEE*

Abstract—Virtualized network slicing allows a multitude of logical networks to be created on a common substrate infrastructure to support diverse services. A virtualized network slice is a logical combination of multiple virtual network functions, which run on virtual machines (VMs) as software applications by virtualization techniques. As the performance of network slices hinges on the normal running of VMs, detecting and analyzing anomalies in VMs are critical. Based on the three-tier management framework of virtualized network slicing, we first develop a federated learning (FL) based three-tier distributed VM anomaly detection framework, which enables distributed network slice managers to collaboratively train a global VM anomaly detection model while keeping metrics data locally. The high-dimensional, imbalanced, and distributed data features in virtualized network slicing scenarios invalidate the existing anomaly detection models. Considering the powerful ability of generative adversarial network (GAN) in capturing the distribution from complex data, we design a new multi-discriminator Bidirectional Wasserstein GAN with Gradient Penalty (BiWGAN-GP) model to learn the normal data distribution from high-dimensional resource metrics datasets that are spread on multiple VM monitors. The multi-discriminator BiWGAN-GP model can be trained over distributed data sources, which avoids high communication and computation overhead caused by the centralized collection and processing of local data. We define an anomaly score as the discriminant criterion to quantify the deviation of new metrics data from the learned normal distribution to detect abnormal behaviors arising in VMs. The efficiency and effectiveness of the proposed collaborative anomaly detection algorithm are validated through extensive experimental evaluation on a real-world dataset.

Index Terms—Virtualized network slicing, virtual machines, collaborative anomaly detection, federated learning (FL), bidirectional Wasserstein generative adversarial network with gradient penalty (BiWGAN-GP).

1 INTRODUCTION

N ETWORK slicing allows network operators to build multiple self-contained logical networks over a common underlying network infrastructure to accommodate the wide range of service requirements [1], [2]. The rise of network function virtualization (NFV) and software defined networks (SDN) facilitates the deployment of network slices through running classical network functions as software applications in virtual machines (VMs) instead of dedicated hardware [3]. However, with the increasingly extensive application of VMs, their security and stability issues have drawn a wide attention [4]. As the performance of network slices hinges on the normal running of VMs, it is crucial to detect anomalies in VMs in a timely manner to ensure the service quality of network slices.

Anomaly detection is necessary to help find and identify abnormal behaviors in VMs before serious failures occur. The abnormal behaviors of a VM are defined as any performance degradation deviating from normal behaviors [5]. Existing researches [6], [7], [8] have shown that the abnormal behaviors of VMs usually come with a significant change in resource metrics, so it is a good way to implement anomaly detection for VMs by collecting and analyzing its multi-dimensional resource metrics data. Although there have been many interesting researches for anomaly detection, including statistical and probability methods [9], [10], distance-based methods [11], [12], domain-based methods [13], [14], reconstruction-based methods [15], [16], [17], and information theory based methods [18], as classified in [19], detecting anomalies of VMs in virtualized network slicing environment still faces many challenges:

Firstly, new and emerging user cases in virtualized network slicing environment have resulted in diverse types of metrics data and highly complex data structures, which are hard to accurately capture with traditional anomaly detection methods.

Secondly, the boundary between normal metrics data and abnormal ones is hard to get for three reasons: 1) Abnormal events rarely occur in real networks, so available normal and abnormal data are extremely imbalanced; 2) It is too costly to obtain enough and various kinds of abnormal metrics data in the environment with multiple demanddiverging virtual networks; 3) Novel anomalies often arise in virtualized network slicing environment.

Thirdly, substrate networks consist of multiple regions and VMs that constitute a network slice can cross multiple

W. Wang, C. Liang, L. Tang and Q. Chen are with the School of Communication and Information Engineering and the Key Laboratory of Mobile Communication, Chongqing University of Posts and Telecommunications, Chongqing 400065, China (e-mail: 1961797154@qq.com; liangcc@cqupt.edu.cn; tangluncq@163.com; cqb@cqupt.edu.cn).

H. Yanikomeroglu is with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada (e-mail: halim@sce.carleton.ca).

regions, so resource metrics data of VMs are distributed in the networks. Existing works commonly use a centralized database to aggregate all local metrics data, such as Gnocchi [20] and Prometheus [21], [22], which will compromise the communication and computation efficiency when learning a global VM anomaly detection model.

Generative adversarial network (GAN) has drawn substantial attention for its powerful ability in capturing the distribution from high-dimensional real-world data [23]. To conquer the difficulties in capturing complex data structures, finding boundaries between normal and abnormal data and high cost of centralized training, we design a new multi-discriminator Bidirectional Wasserstein GAN with Gradient Penalty (BiWGAN-GP) to capture the distribution of normal metrics data in VMs. Compared to GAN, the multi-discriminator BiWGAN-GP includes four main improvements: 1) Replace Jensen-Shannon (JS) divergence with Wasserstein-1 distance to avoid gradient vanishing happened in GAN commonly (Wasserstein GAN (WGAN) [24]); 2) Besides a generator and a discriminator, an encoder is added to realize the reverse mapping of generator to facilitate the establishment of discriminant criterion for anomaly detection (Bidirectional GAN (BiGAN) [25]); 3) Gradient penalty is used to enforce the Lipschitz constraint in WGAN to solve the optimization difficulties of weight clipping (WGAN-GP [26]); 4) Instead of a centralized discriminator, multiple discriminators are running locally to avoid high communication and computation overhead caused by the centralized collection and processing of local data.

The management framework for virtualized network slicing is a three-tier structure, including virtualized infrastructure managers (VIMs) for VMs, network slice managers for network slices, and network controller for the whole networks. For adapting to the three-tier management framework of virtualized network slicing, we develop a federated learning (FL) based three-tier distributed anomaly detection framework to identify abnormal behaviors arising in VMs. The introduction of FL enables to collaboratively train a global VM anomaly detection model over multiple distributed datasets [27]. The novelty of the paper lies in the combination of multi-discriminator BiWGAN-GP and FL to detect anomalies in VMs in the context of virtualized network slicing environment. Specifically, the main contributions of the paper are summarized as follows:

- To lower the latency when collecting data, separate databases are deployed for different regions. Besides, to ensure the isolation among different network slices, in each region, we deploy a VM monitor for a network slice to collect and store metrics data of VMs in the network slice as a local database.
- Considering the three-tier management framework of virtualized network slicing, we develop an FL-based three-tier distributed anomaly detection framework to identify abnormal behaviors arising in VMs. The developed framework enables to collaboratively train a global VM anomaly detection model while keeping metrics data locally through the hierarchical cooperation among VM monitors, network slice managers and the network controller.
- In view of the high-dimensional, imbalanced, and

distributed data features in virtualized network slicing scenarios, we propose a new multi-discriminator BiWGAN-GP algorithm to learn the normal data distribution from high-dimensional metrics datasets that are spread on multiple VM monitors. The proposed algorithm deploys multiple discriminators running locally in VM monitors to avoid the centralized collection and processing of local data. In addition, an anomaly score is defined as the discriminant criterion to quantify the deviation of new metrics data from the learned normal distribution to detect abnormal behaviors arising in VMs.

• We implement extensive simulations on a real-world dataset to evaluate the efficiency and effectiveness of the proposed federated multi-discriminator BiWGAN-GP based collaborative anomaly detection algorithm. Experimental results demonstrate that the proposed algorithm can accurately detect abnormal behaviors in VMs with low communication and computation cost.

The rest of the paper is organized as follows. Section 2 presents the system model of virtualized network slicing management and the corresponding distributed anomaly detection framework. Section 3 introduces different GAN algorithms and proposes the multi-discriminator BiWGAN-GP algorithm. In Section 4, we present the training and detection processes of the three-tier FL-based distributed anomaly detection framework. The performance of the developed framework is evaluated in Section 5. Then, we provide a discussion on important related works in Section 6 and finally Section 7 concludes the paper.

2 SYSTEM MODEL

2.1 Three-tier management framework

The network slice management and orchestration (MANO) is responsible for the control of virtualized network slicing [3]. The system model of virtualized network slicing management is shown in Fig. 1, which is a three-tier structure. The specific roles of each tier are described as follows:

Substrate networks: Substrate networks are divided into N regions, where each region consists of various physical nodes including servers, switches and gateways. Physical nodes provide basic storage, computing and network resources required by the creation and operation of VMs. One physical node can host multiple VMs supported by the virtualization layer. VIM is responsible for the management and maintenance of each region, whose implementations include VIM connector developed in Open-Source MANO project [29], OpenStack and Docker.

Network slice instances: Network slices are defined as logical networks running on a common underlying infrastructure, mutually isolated, created on demand and with independent control and management [1], which is realized by distributed network slice managers. Network slice managers can be implemented by Juju charms [29], Tacker [30] or Puppet [3] solutions. A virtualized network slice consists of a set of virtual network functions and virtual links connecting these functions. Virtual network functions are often running on VMs as software applications [31].



Fig. 1. System model of virtualized network slicing management.



Fig. 2. FL-based three-tier anomaly detection framework. This framework's workflow consists of seven steps as follows: (1) The VM monitor trains its own local discriminator model based on the local training dataset; (2) The VM monitor uploads error feedbacks for generator and encoder to its network slice manager; (3) The network slice manager updates its generator and encoder models according to the error feedbacks from all connected VM monitors, then generates fake data based on the updated generator and encoder; (4) The network slice manager sends the fake data to update its local discriminators on VM monitors; 5 After some iterations, all distributed network slice managers upload their generator and encoder models to the network controller for global aggregation; (6) The network controller obtains global generator and encoder models by the FedAVG algorithm [28]; 7 The network controller sends new global generator and encoder models to distributed network slice managers for their further updates. The above steps are implemented repeatedly until global models converge.

Service instances: In service instance layer, service requests are received and translated into required network functions. The network controller is responsible for the orchestration of different network slices through coordinating the available resources of the whole networks, whose

functionalities can be provided by OSM [29], OpenStack Tacker [30] and OpenBaton [32].

2.2 Three-tier distributed anomaly detection framework

Based on the system model of virtualized network slicing management, we develop a three-tier distributed anomaly detection framework, as illustrated in Fig. 2, to detect abnormal behaviors of VMs in virtualized network slicing environment. As the abnormal behaviors of VMs usually come with significant changes in resource metrics, it is a good way to implement anomaly detection by collecting and analyzing their resource metrics data. We design a multidiscriminator BiWGAN-GP algorithm as the basic method for anomaly detection, which is elaborated in Section 3.

To lower the latency when collecting data and to ensure the isolation among network slices, in each region, we deploy a VM monitor for a network slice to collect and store metrics data of VMs in the network slice as a local database. VM monitors resort to the virtualization layer to capture resource metrics of VMs [4], [6]. The resource metrics are related with the CPU usage, memory consumption, disk read/write and network input/output. To train a global VM detection model, the resource metrics data from all VMs are required. To avoid the centralized collection and processing of local data, we design an FL-based three-tier distributed anomaly detection framework to collaboratively train a global VM anomaly detection model over datasets that are spread on multiple VM monitors. The framework's workflow is illustrated in Fig. 2.

The detailed training and detection processes for the FLbased three-tier distributed anomaly detection framework are presented in Section 4.

3 MULTI-DISCRIMINATOR BIWGAN-GP

The multi-discriminator BiWGAN-GP algorithm, which is a distributed form of BiWGAN-GP, trains local discriminators on local VM monitors using their own training datasets. The

multi-discriminator BiWGAN-GP is designed to capture the distribution of normal metrics data in VMs.

3.1 Existing models: GAN, BiGAN, WGAN-GP

GAN: GAN [33] consists of two models: generator *G* and discriminator *D*. Given the real metrics data $\{\mathbf{X}^r\}_{r=1}^{N_{td}}$ as the training dataset, *G* tries to transform a low-dimensional random noise \mathbf{z} extracted from a known distribution $P_{\mathbf{z}}$ into a fake data $\bar{\mathbf{X}}$ (namely $\bar{\mathbf{X}} = G(\mathbf{z})$), such that *D* cannot distinguish $\bar{\mathbf{X}}$ from real data $\{\mathbf{X}^r\}_{r=1}^{N_{td}}$. Let $P_{\mathbf{X}}$ and $P_{\bar{\mathbf{X}}}$ denote the distributions of real and fake data, and $D(\mathbf{X})$ denote the probability that \mathbf{X} obeys the distribution of real data. The training objective for GAN is defined as [33]

$$\min_{G} \max_{D} V(G, D)$$

$$= \mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{X}}} [\log(D(\mathbf{X}))] + \mathop{\mathbb{E}}_{\mathbf{z} \sim P_{\mathbf{z}}} [\log(1 - D(G(\mathbf{z})))]$$

$$= \mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{X}}} [\log(D(\mathbf{X}))] + \mathop{\mathbb{E}}_{\bar{\mathbf{X}} \sim P_{\bar{\mathbf{X}}}} [\log(1 - D(\bar{\mathbf{X}}))].$$

$$(1)$$

G and *D* are updated alternately. If an optimal *D* is obtained, the training process of *G* theoretically leads to minimizing the JS divergence between P_X and $P_{\bar{X}}$, namely,

$$\min_{G} V(G) = 2D_{JS}(P_{\mathbf{X}} || P_{\bar{\mathbf{X}}}) - \log 4,$$
 (2)

where $D_{JS}(\cdot || \cdot)$ represents the JS divergence. When G and D are both trained into optimality, $P_{\mathbf{X}} = P_{\bar{\mathbf{X}}}$ can be achieved, which indicates that G has captured the distribution of real data.

BiGAN: Besides *G* and *D* in classical GAN, BiGAN also contains an encoder *E*. GAN has realized a mapping from low-dimensional random noise *z* in latent space to fake data \bar{X} in data space ($\bar{X} = G(z)$), but the reverse process is unavailable. The encoder *E* in BiGAN completes the mapping from the real data *X* in data space to a lowdimensional representation *f* in latent space (f = E(X)). *D* in BiGAN tries to distinguish between the joint pairs (X, E(X)) and (G(z), z) in both data space and latent space. The training objective for BiGAN is given by

$$\min_{G,E} \max_{D} V(G, E, D)$$

$$= \mathop{\mathbb{E}}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}} [\log(D(\boldsymbol{X}, E(\boldsymbol{X})))] + \mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_{\boldsymbol{z}}} [\log(1 - D(G(\boldsymbol{z}), \boldsymbol{z}))]$$

$$= \mathop{\mathbb{E}}_{(\boldsymbol{X}, \boldsymbol{f}) \sim P_{\boldsymbol{X}\boldsymbol{f}}} [\log(D(\boldsymbol{X}, \boldsymbol{f})] + \mathop{\mathbb{E}}_{(\bar{\boldsymbol{X}}, \boldsymbol{z}) \sim P_{\bar{\boldsymbol{X}}\boldsymbol{z}}} [\log(1 - D(\bar{\boldsymbol{X}}, \boldsymbol{z}))],$$
(3)

where $P_{\boldsymbol{X}\boldsymbol{f}}$ and $P_{\bar{\boldsymbol{X}}\boldsymbol{z}}$ represent the joint distributions of $(\boldsymbol{X}, \boldsymbol{f})$ and $(\bar{\boldsymbol{X}}, \boldsymbol{z})$, respectively. Given an optimal D, the training objective of G and E is also to minimize the JS divergence between $P_{\boldsymbol{X}\boldsymbol{f}}$ and $P_{\bar{\boldsymbol{X}}\boldsymbol{z}'}$ namely,

$$\min_{G,E} V(G,E) = 2D_{JS}(P_{\boldsymbol{X}\boldsymbol{f}}||P_{\bar{\boldsymbol{X}}\boldsymbol{z}}) - \log 4.$$
(4)

The global minimum of (4) can be achieved when $P_{\boldsymbol{X}\boldsymbol{f}} = P_{\bar{\boldsymbol{X}}\boldsymbol{z}}$ is satisfied. By this time, E and G are each other's inverse, namely $G(E(\boldsymbol{X})) = \boldsymbol{X}$ and $E(G(\boldsymbol{z})) = \boldsymbol{z}$ [25].

WGAN-GP: In the early phase of GAN training, the very little overlap between P_X and $P_{\bar{X}}$ will cause their JS divergence being a constant and the gradient vanishing, which will terminate the training exceptionally. To solve it,



Fig. 3. Multi-discriminator BiWGAN-GP framework.

WGAN [24] replaces the JS divergence with Wasserstein-1 distance, which is computed by

$$W_1(P_{\boldsymbol{X}}||P_{\bar{\boldsymbol{X}}}) = \sup_{f \in \mathbb{L}_1} \{ \mathbb{E}_{\boldsymbol{X} \sim P_{\boldsymbol{X}}}[f(\boldsymbol{X})] - \mathbb{E}_{\bar{\boldsymbol{X}} \sim P_{\bar{\boldsymbol{X}}}}[f(\bar{\boldsymbol{X}})] \}, \quad (5)$$

where \mathbb{L}_1 is the set of 1-Lipschitz functions, which is satisfied through weight clipping [24]. Due to the optimization difficulties of weight clipping, [26] proposed WGAN with gradient penalty (WGAN-GP) to enforce the Lipschitz constraint. WGAN-GP uses *D* with the norm of its gradient penalty to approximate the function $f \in \mathbb{L}_1$ in WGAN. Therefore, *D* is trained based on the following objective:

$$\max_{D} V(D) = \mathop{\mathbb{E}}_{\mathbf{X} \sim P_{\mathbf{X}}} [D(X)] - \mathop{\mathbb{E}}_{\bar{\mathbf{X}} \sim P_{\bar{\mathbf{X}}}} [D(\bar{\mathbf{X}})] + \eta \mathop{\mathbb{E}}_{\bar{\mathbf{X}} \sim P_{\bar{\mathbf{X}}}} [(||\nabla_{\widehat{\mathbf{X}}} D(\widehat{\mathbf{X}})||_2 - 1)^2],$$
(6)

where η is the penalty coefficient and \widehat{X} is defined to uniformly sample along straight lines between pairs of points extracted from the distributions $P_{\mathbf{X}}$ and $P_{\overline{\mathbf{X}}}$. Therefore, $\widehat{\mathbf{X}}$ is the weighted sum of \mathbf{X} and $\overline{\mathbf{X}}$: $\widehat{\mathbf{X}} = \varepsilon \mathbf{X} + (1 - \varepsilon)\overline{\mathbf{X}}$, where $\varepsilon \in U[0, 1]$. *G* is trained through

$$\min_{G} V(G) = -\mathop{\mathbb{E}}_{\bar{\boldsymbol{X}} \sim P_{\bar{\boldsymbol{X}}}} [D(\bar{\boldsymbol{X}})] = -\mathop{\mathbb{E}}_{\boldsymbol{z} \sim P_{\boldsymbol{z}}} [D(G(\boldsymbol{z}))].$$
(7)

3.2 BiWGAN-GP and its distributed form

We propose BiWGAN-GP to combine the strengths of BiGAN and WGAN-GP. After obtaining $(\boldsymbol{X}, \boldsymbol{f})$ and $(\bar{\boldsymbol{X}}, \boldsymbol{z})$, $(\widehat{\boldsymbol{X}}, \widehat{\boldsymbol{z}})$ is calculated through $(\widehat{\boldsymbol{X}}, \widehat{\boldsymbol{z}}) = \varepsilon(\boldsymbol{X}, \boldsymbol{f}) + (1 - \varepsilon)(\bar{\boldsymbol{X}}, \boldsymbol{z})$ similar to the calculation of $\widehat{\boldsymbol{X}}$. In BiWGAN-GP, D is trained through

$$\max_{D} V(D) = \mathop{\mathbb{E}}_{(\boldsymbol{X},\boldsymbol{f})\sim P_{\boldsymbol{X}\boldsymbol{f}}} [D(\boldsymbol{X},\boldsymbol{f})] - \mathop{\mathbb{E}}_{(\bar{\boldsymbol{X}},\boldsymbol{z})\sim P_{\bar{\boldsymbol{X}}\boldsymbol{z}}} [D(\boldsymbol{X},\boldsymbol{z})] + \eta \mathop{\mathbb{E}}_{(\widehat{\boldsymbol{X}},\widehat{\boldsymbol{z}})\sim P_{\widehat{\boldsymbol{X}}\widehat{\boldsymbol{z}}}} [(||\nabla_{(\widehat{\boldsymbol{X}},\widehat{\boldsymbol{z}})}D((\widehat{\boldsymbol{X}},\widehat{\boldsymbol{z}}))||_{2} - 1)^{2}],$$
(8)

where the definition of $P_{\widehat{X}\widehat{z}}$ is similar to $P_{\widehat{X}}$. After training D, the Wasserstein-1 distance $W_1(P_{Xf}||P_{\overline{X}z})$ between P_{Xf} and $P_{\overline{X}z}$ can be expressed as

$$W_1(P_{\boldsymbol{X}\boldsymbol{f}}||P_{\bar{\boldsymbol{X}}\boldsymbol{z}}) = \\ \underset{(\boldsymbol{X},\boldsymbol{f})\sim P_{\boldsymbol{X}\boldsymbol{f}}}{\mathbb{E}} [D(\boldsymbol{X},\boldsymbol{f})] - \underset{(\bar{\boldsymbol{X}},\boldsymbol{z})\sim P_{\bar{\boldsymbol{X}}\boldsymbol{z}}}{\mathbb{E}} [D(\bar{\boldsymbol{X}},\boldsymbol{z})].$$
⁽⁹⁾

The training objective of *G* and *E* is to minimize $W_1(P_{\boldsymbol{X}\boldsymbol{f}}||P_{\boldsymbol{\bar{X}}\boldsymbol{z}})$, namely,

$$\min_{G,E} V(G,E) = W_1(P_{\boldsymbol{X}\boldsymbol{f}}||P_{\bar{\boldsymbol{X}}\boldsymbol{z}}).$$
(10)

A multi-discriminator BiWGAN-GP is designed to learn the distribution of all metrics data within a network slice. In multi-discriminator BiWGAN-GP, a pair of generator and encoder is trained on a network slice manager and Ndiscriminators are trained on VM monitors.

The multi-discriminator BiWGAN-GP framework is illustrated in Fig. 3. The network slice manager selects and sends the generated data to each VM monitor, and the discriminators in VM monitors try to distinguish generated data from local real data. Therefore, the optimization objective of multi-discriminator BiWGAN-GP is defined as

$$\min_{G,E} \max_{D} V(G, E, D) = \frac{1}{N} \sum_{n=1}^{N} \{ \mathop{\mathbb{E}}_{(\boldsymbol{X},\boldsymbol{f})\sim P_{\boldsymbol{X}\boldsymbol{f}}} [D_n(\boldsymbol{X},\boldsymbol{f})] - \mathop{\mathbb{E}}_{(\boldsymbol{\bar{X}},\boldsymbol{z})\sim P_{\boldsymbol{\bar{X}}\boldsymbol{z}}} [D_n(\boldsymbol{\bar{X}},\boldsymbol{z})] \\ + \mathop{\mathbb{E}}_{(\boldsymbol{\widehat{X}},\boldsymbol{\widehat{z}})\sim P_{\boldsymbol{\widehat{X}}\boldsymbol{\widehat{z}}}} [(||\nabla_{(\boldsymbol{\widehat{X}},\boldsymbol{\widehat{z}})} D_n((\boldsymbol{\widehat{X}},\boldsymbol{\widehat{z}}))||_2 - 1)^2] \}.$$
(11)

3.3 Modeling the multi-discriminator BiWGAN-GP

As the available training dataset of metrics is time series, we choose Vanilla Long Short-Term Memory Network (VLSTM) as the basic model of generator *G* and encoder *E*, which has shown superior performance on time-series-related tasks as a kind of recurrent neural networks [34]. Given $[y_1, ..., y_t]$ as input, then for each time epoch τ ($\tau = 1, ..., t$), we have

$$\begin{aligned} & \boldsymbol{i}_{\tau} = \sigma(\boldsymbol{w}_{i}.[\boldsymbol{y}_{\tau}, \ \boldsymbol{h}_{\tau-1}, \ \boldsymbol{c}_{\tau-1}] + \boldsymbol{b}_{i}), \\ & \boldsymbol{f}_{\tau} = \sigma(\boldsymbol{w}_{f}.[\boldsymbol{y}_{\tau}, \ \boldsymbol{h}_{\tau-1}, \ \boldsymbol{c}_{\tau-1}] + \boldsymbol{b}_{f}), \\ & \boldsymbol{c}_{\tau} = \boldsymbol{f}_{\tau} \odot \boldsymbol{c}_{\tau-1} + \boldsymbol{i}_{\tau} \odot \tanh(\boldsymbol{w}_{z}.[\boldsymbol{y}_{\tau}, \ \boldsymbol{h}_{\tau-1}] + \boldsymbol{b}_{z}), \quad (12) \\ & \boldsymbol{o}_{\tau} = \sigma(\boldsymbol{w}_{o}.[\boldsymbol{y}_{\tau}, \ \boldsymbol{h}_{\tau-1}, \ \boldsymbol{c}_{\tau}] + \boldsymbol{b}_{o}), \\ & \boldsymbol{h}_{\tau} = \boldsymbol{o}_{\tau} \odot \tanh(\boldsymbol{c}_{\tau}), \end{aligned}$$

where i_{τ} , f_{τ} and o_{τ} are the input gate, forget gate and output gate, respectively. c_{τ} denotes the memory unit and h_{τ} denotes the hidden state. w_* and b_* represent weights and bias, respectively. The symbol $\sigma(\cdot)$ represents the sigmoid function and \odot denotes the operation of elementwise multiplication.

Build G: We build the generator *G* into a three-layer structure, including two VLSTM layers and one fully connected layer activated by linear. In network slice managers, after extracting z from P_z , *G* maps z into the fake data \bar{X}

$$\bar{\boldsymbol{X}} = G(\boldsymbol{z}; \ [\boldsymbol{w}_G, \boldsymbol{b}_G]), \tag{13}$$

where w_G and b_G denote weights and biases of G, which compose model parameters θ_G .

Build E: As encoder E is the inverse function of G, it is also a three-layer structure with two VLSTM layers and one fully connected layer activated by linear, but they are grouped in a fully reverse order to G. E maps the real data X to a low-dimensional representation f

$$\boldsymbol{f} = E(\boldsymbol{X}; \ [\boldsymbol{w}_E, \boldsymbol{b}_E]), \tag{14}$$

where w_E and b_E denote weights and biases of E, which compose model parameters θ_E .

Build D_n : We build the discriminator D_n $(n \in N)$ with three fully connected layer, where the last layer is sigmoid activated to get a probability in range (0, 1). The discriminator D_n in each VM monitor maps $(\mathbf{X}_n, \mathbf{f}_n)$ and $(\bar{\mathbf{X}}_n, \mathbf{z}_n)$ to a scalar d

$$d = D_n \left\{ \left[(\boldsymbol{X}_n, \boldsymbol{f}_n), \ (\bar{\boldsymbol{X}}_n, \boldsymbol{z}_n) \right]; \left[\boldsymbol{w}_{D_n}, \boldsymbol{b}_{D_n} \right] \right\},$$
(15)

where w_{D_n} and b_{D_n} denote weights and biases of D_n , which compose model parameters θ_{D_n} .

The training of multi-discriminator BiWGAN-GP algorithm for a network slice is iteratively performed by VM monitors and the network slice manager. Next, we separately specify the training of D_n ($n \in N$) on each VM monitor and the training of G and E on the network slice manager.

Training on VM monitors: Each VM monitor hosts a discriminator D_n with its parameters θ_{D_n} . In each iteration, a VM monitor contains two functions: 1) Receive the generated data to update its discriminator; 2) Calculate the error feedbacks for generator and encoder updates.

1) Discriminator Update: Each VM monitor receives the generated data pairs (\bar{X}_n, z_n) and the low-dimensional representation f_n of X_n from the network slice manager, whose batch sizes are all M. Then, the VM monitor performs K training iterations on D_n . In the kth local iteration, for each batch (X_n^m, f_n^m) and (\bar{X}_n^m, z_n^m) in (X_n, f_n) and (\bar{X}_n, z_n) , the loss function of D_n is calculated by

$$L_{D_n}^{m} = -\{D_n(\boldsymbol{X}_n^{m}, \boldsymbol{f}_n^{m}) - D_n(\bar{\boldsymbol{X}}_n^{m}, \boldsymbol{z}_n^{m}) + \eta[||\Delta_{(\widehat{\boldsymbol{X}}_n^{m}, \widehat{\boldsymbol{z}}_n^{m})} D_n(\widehat{\boldsymbol{X}}_n^{m}, \widehat{\boldsymbol{z}}_n^{m})||_2 - 1]^2\}.$$
(16)

Accordingly, the gradients $\Delta \theta_{D_n}$ are calculated by

$$\Delta \theta_{D_n} = \frac{1}{M} \sum_{m=1}^{M} \frac{\partial L_{D_n}^m}{\partial \theta_{D_n}}.$$
(17)

We update the parameters θ_{D_n} of discriminator D_n using the Adam optimizer method

$$\theta_{D_n} \leftarrow \operatorname{Adam}(\Delta \theta_{D_n}, \theta_{D_n}, \alpha, \beta_1, \beta_2), \tag{18}$$

where α , β_1 and β_2 are hyperparameters of Adam optimizer.

2) Error Feedbacks Calculation: To update the parameters of G and E in the network slice manager, each VM monitor calculates the error feedbacks F_G^n and F_E^n based on the local loss function L_{EG}^n and D_n after K training iterations. The local loss function L_{EG}^n for G and E in each VM monitor is computed as

$$L_{EG}^{n} = \frac{1}{M} \sum_{m=1}^{M} \{ D_{n}(\boldsymbol{X}_{n}^{m}, \boldsymbol{f}_{n}^{m}) - D_{n}(\bar{\boldsymbol{X}}_{n}^{m}, \boldsymbol{z}_{n}^{m}) \}.$$
(19)

The error feedback F_E^n of E consists of M vectors $\{e_n^1, ..., e_n^M\}$, where e_n^m is defined as

$$e_n^m = \frac{\partial L_{EG}^n}{\partial (\boldsymbol{X}_n^m, \boldsymbol{f}_n^m)}.$$
(20)

Similarly, the error feedback F_G^n of G also consists of M vectors $\{\boldsymbol{g}_n^1, ..., \boldsymbol{g}_n^M\}$, where \boldsymbol{g}_n^m is defined as

$$g_n^m = \frac{\partial L_{EG}^n}{\partial (\bar{\boldsymbol{X}}_n^m, \boldsymbol{z}_n^m)}.$$
(21)

 F_E^n and F_G^n obtained in each VM monitor will be sent to the network slice manager for G and E updates.

Training on the network slice manager: The network slice manager hosts G and E, which contains two functions in each iteration: 1) Generate and encode data using the generator and encoder; 2) Receive error feedbacks to update the generator and encoder.

1) Data Generation: The network slice manager first receives the set of training batches $\{X_n\}_{n=1}^N$ from N VM monitors. E turns $\{X_n\}_{n=1}^N$ into a set of low-dimensional representations $\{f_n\}_{n=1}^N$. In addition, G transforms a set of low-dimensional random noise $\{\boldsymbol{z}_n\}_{n=1}^N$ extracted from a known distribution P_z into a set of fake data $\{X_n\}_{n=1}^N$. Then, the network slice manager sends f_n , z_n and \bar{X}_n to each VM monitor to train D_n ($n \in N$).

2) G and E Updates: After receiving error feedbacks F_E^n and F_G^n from each VM monitor, the network slice manager updates parameters θ_G and θ_E of G and E. We use the averaging operation to aggregate error feedbacks from all VM monitors as it is the most common method to merge feedbacks updated in parallel [35]. Specifically, the gradients $\Delta \theta_G$ of θ_G are deduced from feedbacks F_G^n ($n \in N$) by

$$\Delta \theta_G^l = \frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \frac{\partial L_{EG}^n}{\partial \theta_G^l} = \frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \frac{\partial L_{EG}^n}{\partial (\bar{\boldsymbol{X}}_n^m, \boldsymbol{z}_n^m)} \frac{\partial (\bar{\boldsymbol{X}}_n^m, \boldsymbol{z}_n^m)}{\partial \theta_G^l} \qquad (22) = \frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \boldsymbol{g}_n^m \frac{\partial (\bar{\boldsymbol{X}}_n^m, \boldsymbol{z}_n^m)}{\partial \theta_G^l},$$

where $\Delta \theta_G^l$ is the *l*-th $(l \in L)$ element of gradients $\Delta \theta_G$. After the gradients are calculated, the network slice manager updates parameters θ_G using Adam optimizer

$$\theta_G \leftarrow \operatorname{Adam}\{(\Delta \theta_G^1, ..., \Delta \theta_G^L), \theta_G, \alpha, \beta_1., \beta_2\}.$$
 (23)

Similarly, the gradients $\Delta \theta_E$ of parameters θ_E are deduced from all feedbacks F_E^n ($n \in N$) by

$$\Delta \theta_E^l = \frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \frac{\partial L_{EG}^n}{\partial \theta_E^l}$$

= $\frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \frac{\partial L_{EG}^n}{\partial (\boldsymbol{X}_n^m, \boldsymbol{f}_n^m)} \frac{\partial (\boldsymbol{X}_n^m, \boldsymbol{f}_n^m)}{\partial \theta_E^l}$ (24)
= $\frac{1}{MN} \sum_{n=1}^N \sum_{m=1}^M \boldsymbol{e}_n^m \frac{\partial (\boldsymbol{X}_n^m, \boldsymbol{f}_n^m)}{\partial \theta_E^l}$,

where $\Delta \theta_E^l$ is the *l*-th ($l \in L$) element of gradients $\Delta \theta_E$. As G and E are each other's inverse, $\Delta \theta_E$ has the same number of elements with $\Delta \theta_G$. Then, the network slice manager updates parameters θ_E using Adam optimizer

$$\theta_E \leftarrow \operatorname{Adam}\{(\Delta \theta_E^1, ..., \Delta \theta_E^L), \theta_E, \alpha, \beta_1, \beta_2\}.$$
 (25)

The detailed steps of multi-discriminator BiWGAN-GP algorithm are presented in Algorithm 1.

4 FL-BASED THREE-TIER DISTRIBUTED ANOMALY **DETECTION FRAMEWORK**

The proposed multi-discriminator BiWGAN-GP algorithm is adapted to the management of VMs within a network

Algorithm 1 Multi-discriminator BiWGAN-GP algorithm

- **Hyperparameters:** The number of iterations K between two E and G iterations. Batch size M. Adam hyperparameters α , β_1 , β_2 . The prior distribution P_z . The number of training iterations *I*.
- **Input:** Initialized model parameters: θ_G , θ_E , θ_{D_n} $(n \in N)$, and distributed training datasets $\bigcup_{n=1}^{N} \{X_n^r\}_{r=1}^{N_{td}}$.
- **Output:** Converged model parameters: θ_G , θ_E , θ_{D_n} ($n \in$ N).
- 1: for i = 1 : I do
- 2: \triangleright Update D_n ($n \in N$) on VM monitors
- Sample X_n from its own training dataset $\{X_n^r\}_{r=1}^{N_{td}}$ 3: and send it to the network slice manager
- Receive (X_n, z_n) and f_n from the network slice 4: manager to obtain the data pairs (X_n, f_n) and $(\boldsymbol{X}_n, \boldsymbol{z}_n)$
- for k = 1 : K do 5:
- for $m = 1: M \operatorname{do}$ 6:
- Calculate $(\widehat{X}_n^m, \widehat{z}_n^m) = \varepsilon(X_n^m, f_n^m) + (1 \varepsilon)$ 7: $\varepsilon)(oldsymbol{X}_n^m,oldsymbol{z}_n^m)$ 8:
 - Calculate $L_{D_m}^m$ according to (16)
 - if k == K then
- 10: Calculate L_{EG}^n according to (19)
- Calculate elements e_n^m and g_n^m of error 11: feedbacks according to (20) and (21)

12: end if 13: end for

9:

14:

- Update gradients $\Delta \theta_{D_n}$ and D_n 's parameters θ_{D_n} according to (17) and (18)
- 15: end for
- Send the error feedbacks $F_G^n = \{\boldsymbol{g}_n^1, ..., \boldsymbol{g}_n^M\}$ and 16: $F_E^n = \{e_n^1, ..., e_n^M\}$ to the network slice manager
- 17: \triangleright Update *E* and *G* on the network slice manager
- Receive a set of training batches $\{X_n\}_{n=1}^N$ from N VM 18: monitors
- 19:
- Sample a set of noise $\{\boldsymbol{z}_n\}_{n=1}^N \sim P_{\boldsymbol{z}}$ Obtain $\{\boldsymbol{f}_n\}_{n=1}^N \leftarrow E(\{\boldsymbol{X}_n\}_{n=1}^N)$ and $\{\bar{\boldsymbol{X}}_n\}_{n=1}^N \leftarrow$ 20: $G(\{\boldsymbol{z}_n\}_{n=1}^N)$
- Send f_n , z_n and \bar{X}_n to each VM monitor 21:
- Deduce gradients $\Delta \theta_G$ of G from all feedbacks F_G^n 22: $(n \in N)$ based on (22), then update G's parameters θ_G according to (23)
- 23: Deduce gradients $\Delta \theta_E$ of E from all feedbacks F_E^n $(n \in N)$ based on (24), then update E's parameters θ_E according to (25)

slice. To extend it to the entire network, FL is introduced to the multi-discriminator BiWGAN-GP algorithm to develop an FL-based three-tier distributed anomaly detection framework as shown in Fig. 4. From the perspective of a network slice, local training data in each VM monitor are utilized to train the multi-discriminator BiWGAN-GP model. After some iterations, network slice managers will send the local models of generators and encoders to the network controller for global aggregation, such that we can achieve a global VM anomaly detection model through the hierarchical cooperation among VM monitors, network slice managers and the network controller.

^{24:} end for



Fig. 4. FL-based three-tier distributed anomaly detection framework.

4.1 Training

The training of FL-based multi-discriminator BiWGAN-GP algorithm on network slices and the network controller are separately specified as follows:

Network slices: A multi-discriminator BiWGAN-GP model is implemented within a network slice, which contains one pair of generator and encoder and Ndiscriminators. N Discriminators are trained on VM monitors and local generators G_s and encoders E_s ($s \in S$) are trained on network slice managers, where S represents both the number and set of network slices. Network slice managers first receive initial global models G^{global} and E^{global} of generator and encoder from the network controller as the initial local ones. In a network slice s, the network slice manager and VM monitors exchange data and parameters every K iterations for the training of D_n ($n \in N$), G_s and E_s . After L iterations, each network slice manager sends model parameters θ_{G_s} and θ_{E_s} to the network controller for global aggregation. After the averaging operation in network controller, it sends new global models G^{global} and E^{global} to network slice managers for updating local G_s and E_s $(s \in S)$ using the Adam optimizer method

$$\theta_{G_s} \leftarrow \operatorname{Adam}(\Delta \theta_{G_s}, \theta_G^{global}, \alpha, \beta_1, \beta_2), \\ \theta_{E_s} \leftarrow \operatorname{Adam}(\Delta \theta_{E_s}, \theta_E^{global}, \alpha, \beta_1, \beta_2),$$
(26)

where θ_G^{global} and θ_E^{global} are global model parameters of G^{global} and E^{global} .

Network controller: Network controller is responsible for the global aggregation of G_s and E_s ($s \in S$). Network controller contains two main functions: (1) Initialize global models and send them to network slice managers; (2) Aggregate all model parameters θ_{G_s} and θ_{E_s} ($s \in S$) uploaded by network slice managers, then average the received parameters and send global models G^{global} and E^{global} back to network slice managers. G^{global} and E^{global} are updated using averaging operation (i.e., FedAVG algorithm [28]), which is given by

$$\theta_G^{global} = \frac{\sum_{s=1}^{S} Q_s \theta_{G_s}}{Q}, \ \theta_E^{global} = \frac{\sum_{s=1}^{S} Q_s \theta_{E_s}}{Q},$$
(27)

where Q_s is the number of training data in network slice s ($s \in S$), and $Q = \sum_{s=1}^{S} Q_s$ is the total number of training data in all network slices.

The above steps are implemented repeatedly until the global models G^{global} and E^{global} reach optimal convergence. In addition, the network environment is not static but time-varying. The trained model may become inaccurate with time. Therefore, the proposed FL-based multi-discriminator BiWGAN-GP algorithm still needs to be retrained periodically on the newly collected data.

The proposed algorithm can be extended to the dynamic network environment. If there are new network slices joining the FL, we can transfer the well-trained model as their initial models to detect possible abnormal behaviors in current period, and then update their anomaly detection model in the next retraining period.

4.2 Detection

After the generator and encoder are well trained, the copy of them will be sent to each VM monitor for VM anomaly detection. To measure the abnormality of VM v $(v \in V)$, we introduce an anomaly score $A(\mathbf{X}_v)$ for the new metrics data \mathbf{X}_v . The anomaly score $A(\mathbf{X}_v)$ is defined as a weighted combination of the reconstruction loss L_{EG} and the discriminator loss L_D , which is calculated by

$$A(\boldsymbol{X}_{v}) = \gamma L_{EG}(\boldsymbol{X}_{v}) + (1 - \gamma)L_{D}(\boldsymbol{X}_{v}), \qquad (28)$$

Algorithm 2 FL-based multi-discriminator BiWGAN-GP

- **Hyperparameters:** The number of iterations K between two E and G iterations. Batch size M. Adam hyperparameters α , β_1 , β_2 . The number of training iterations I. The number of local iterations L before aggregation. The value of *threshold* for anomaly score.
- **Input:** Initialized model parameters: θ_G^{global} , θ_E^{global} , θ_{D_n} $(n \in N)$, and distributed training datasets $\bigcup_{n=1}^{N} \{ \mathbf{X}_n^r \}_{r=1}^{N_{td}}$ for all $s \in S$, new metrics data \mathbf{X}_v of VM v ($v \in V$).
- **Output:** Converged model parameters: θ_G^{global} , θ_E^{global} and θ_{D_n} ($n \in N$) for all $s \in S$, the abnormality of VM v.
- 1: for i = 1 : I do { \triangleright Training process}
- 2: \triangleright Update D_n ($n \in N$) for all $s \in S$ on VM monitors
- 3: Implement steps (3)-(16) in Algorithm 1
- 4: \triangleright Update E_s and G_s ($s \in S$) on network slice managers
- 5: Receive initial global parameters θ_G^{global} and θ_E^{global} from the network controller as initial local parameters θ_{G_s} and θ_{E_s}
- 6: Implement steps (18)-(24) in Algorithm 1
- 7: if $i \mod L == 0$ then
- 8: Send local parameters θ_{G_s} and θ_{E_s} to the network controller for aggregation
- 9: Receive updated global parameters θ_G^{global} and θ_E^{global} and use them update local G_s and E_s models in each network slice *s* according to (26)
- 10: end if
- 11: \triangleright Update global models G^{global} and E^{global} on the network controller
- 12: Initialize global model parameters θ_G^{global} and θ_E^{global} and sends them to all network slice managers
- 13: Update global model parameters θ_G^{global} and θ_E^{global} by averaging θ_{G_s} and θ_{E_s} ($s \in S$) sent from network slice managers every *L* iterations according to (27)
- 14: end for
- 15: For new metrics data X_v of VM v, calculate its anomaly score $A(X_v)$ based on its corresponding G, E and Daccording to (28) { \triangleright **Detection process**}
- 16: if $A(\mathbf{X}_v) > threshold$ then
- 17: VM v is determined as abnormal
- 18: else
- 19: VM v is determined as normal
- 20: end if

where γ is the weighted coefficient, $L_{EG}(\mathbf{X}_v) = ||\mathbf{X}_v - G(E(\mathbf{X}_v))||_1$ and $L_D(\mathbf{X}_v) = \sigma(D(\mathbf{X}_v), E(\mathbf{X}_v), 1)$. In $L_D(\mathbf{X}_v)$, σ is the sigmoid cross entropy, which defines the discriminator's confidence that \mathbf{X}_v obeys the real data distribution [36]. For new metrics data \mathbf{X}_v of VM v, its VM monitor calculates the anomaly score $A(\mathbf{X}_v)$ based on its own discriminator and the received generator and encoder. If $A(\mathbf{X}_v)$ is larger than a preset threshold, VM v will be determined as abnormal.

The detailed steps of FL-based multi-discriminator BiWGAN-GP algorithm are presented in Algorithm 2.

5 PERFORMANCE EVALUATION

In this section, we evaluate the performance of the proposed FL-based multi-discriminator BiWGAN-GP algorithm in terms of efficiency and effectiveness through extensive experimental evaluation, and compare it with GAN, BiGAN, WGAN-GP, centralized BiWGAN-GP, standalone BiWGAN-GP, and multi-discriminator BiWGAN-GP algorithms.

5.1 Experiment setup

We implement the centralized, standalone, distributed and the developed FL-based distributed anomaly detection framework in Python language using Pytorch package on Intel(R) Core(TM) i7-6700H CPU with 16GB RAM. In our implementations, we emulate 1 network controller, 3 network slice managers, and 9 VM monitors, and use socket to realize the information exchange among them.

1) *Datasets*: We choose the VNFDataset (virtual IP Multimedia IP system)¹ as the experimental dataset for performance evaluation. The entire dataset consists of 6 independent subsets of resource metrics data belonging to 6 different VMs. Each subset contains more than 170 thousand records where each record is described by 26 metric features (1st-4th are CPU-based, 5th-12th are disk-based, 13th-19th are memory-based, and 20th-26th are network-based), which are specified in Table 1. We partition each subset into three disjoint datasets: training dataset for model training, validation dataset for *threshold* setting, and test dataset for performance evaluation.

2) Evaluation metrics: To evaluate the performance of the proposed FL-based multi-discriminator BiWGAN-GP algorithm and the contrast algorithms on anomaly detection, the following metrics are considered as the comparison indicators, which can be categorized into two aspects: (1) Efficiency in terms of the communication, computation and memory cost on VM monitors, network slice managers and network controller in the training process; (2) Effectiveness in terms of Precision, Recall, F1-socre and Accuracy, which are formulated based on four fundamental indicators: TN (true negative) represents the number of correctly identified normal behaviors, FN (false negative) indicates the number of incorrectly identified abnormal behaviors as normal ones, FP (false positive) represents the number of incorrectly identified normal behaviors as abnormal ones, and TP (true positive) indicates the number of correctly identified abnormal behaviors.

Precision refers to the ratio of correctly identified abnormal behaviors to the total identified abnormal ones, calculated by

$$Precision = \frac{TP}{TP + FP}.$$
(29)

Recall refers to the ratio of correctly identified abnormal behaviors to the total actual abnormal ones, calculated by

$$Recall = \frac{TP}{TP + FN}.$$
(30)

1. VNFDataset: virtual IP Multimedia IP system, https://www.kagg le.com/imenbenyahia/clearwatervnf-virtual-ip-multimedia-ip-system.

TABLE 1 Metric Features in VNFDataset

Feature types	Feature numbers	Detailed description
CPU-based	4	Including CPU idle percent, wait percent, system percent, stolen percent, etc.
Disk-based	8	Including disk used percent, read/write requests, read/write frequencies (times/s), read/write rates (kbytes/s), etc.
Memory-based	7	Including average loads, usable percent, usable size (MB), free size (MB), total size (MB), etc.
Network-based	7	Including in/out size (bytes/s), in/out errors (/s), in/out packets (/s), dropped packets (/s), etc.

F1-score is the weighted average of the precision and recall, calculated by

$$F1\text{-}score = 2 \times \frac{precison \times recall}{precison + recall}.$$
(31)

Accuracy refers to the ratio of correctly identified behaviors to the total ones, calculated by

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}.$$
 (32)

5.2 Efficiency

Computational complexity: We analyze the computational complexities of the proposed FL-based multi-discriminator BiWGAN-GP algorithm from three different sides, including VM monitors, network slice managers and the network controller, and compare them with the standalone BiWGAN-GP algorithm (only runs in VM monitors), multi-discriminator BiWGAN-GP algorithm (runs in VM monitors and network slice managers), and centralized BiWGAN-GP algorithm (only runs in the network controller), respectively. In the proposed algorithm, the computational complexity on VM monitors mainly relies on the architecture of discriminator D, and the computational complexities on network slice managers and the network controller mainly rely on the architectures of encoder E and generator G. As D, E and G are all built using neural networks, their computational complexities are hard to accurately describe due to many free variables [37]. For simplicity, we assume that the number of floating point operations to process one-batch data in *D*, *E* and *G* is directly proportional to the cardinality of model parameters $|\theta_D|$, $|\theta_E|$ and $|\theta_G|$, respectively. Besides, the number of floating point operations to update parameters of D, E and G is also assumed to be directly proportional to $|\theta_D|$, $|\theta_E|$ and $|\theta_G|$, respectively. Therefore, the detailed computational complexities on VM monitors, network slice managers and the network controller are analyzed as follows:

- *VM monitors*. On each VM monitor, the complexities of computing error feedbacks and updating model parameters of *D* are $O(4M|\theta_D|)$ and $O(4KM|\theta_D|)$ per training iteration. Therefore, the total computational complexity on each VM monitor is $O(4I(1 + K)M|\theta_D|)$, which is much less than $O(4IKM|\theta_D| + 2IM(|\theta_E| + |\theta_G|))$ of the standalone BiWGAN-GP algorithm on each VM monitor. As the multi-discriminator BiWGAN-GP algorithm is part of the proposed algorithm, they have the same computational complexity on VM monitors.
- *Network slice managers.* On each network slice manager, the complexity of encoding and generating onebatch data using *E* and *G* is $O(MN(|\theta_E| + |\theta_G|))$,

and the complexity of updating model parameters of *E* and *G* is also $O(MN(|\theta_E| + |\theta_G|))$. Therefore, the total computational complexity on each network slice manager is $O(2IMN(|\theta_E| + |\theta_G|))$. Similarly, the proposed algorithm and the multi-discriminator BiWGAN-GP algorithm have the same computational complexity on network slice managers.

• *Network controller*. On the network controller, the proposed algorithm only needs to aggregate model parameters of *E* and *G* from all network slice managers every *L* training iterations. Hence the computational complexity on the network controller is $O(S(|\theta_E| + |\theta_G|)I/L)$, which is much less than $O(2SIMN(2K|\theta_D| + |\theta_E| + |\theta_G|))$ of the centralized BiWGAN-GP algorithm on the network controller.

To validate the convergence of BiWGAN-GP model in different training modes, including centralized, standalone, distributed, and the proposed FL-based distributed modes as shown in Fig. 5, the averaged values of discriminator loss and encoder-generator (EG) loss in each iteration are recorded and depicted in Fig. 6. In centralized training mode, the network controller is responsible for collecting metrics data of all VMs and trains the centralized BiWGAN-GP model for anomaly detection. In standalone training mode, the BiWGAN-GP model is trained inside each VM monitor without any data or parameters exchange. In distributed training mode, the multi-discriminator BiWGAN-GP algorithm presented in Algorithm 1 is used for anomaly detection, where the network slice manager is responsible for the training of generator and encoder and VM monitors are responsible for the training of discriminators. The proposed FL-based distributed training mode is the combination of multiple distributed modes, which is elaborated in Algorithm 2. Fig. 6(a), (b), (c) and (d) shows the training processes of BiWGAN-GP model in centralized, standalone, distributed and FL-based distributed modes, respectively. The discriminator loss and EG loss both converge to stable values after about 300 training iterations, which indicates that the generators, encoders and discriminators all can be successfully trained in all four training modes. By contrast, the proposed FL-based distributed training mode enables the discriminator loss and EG loss to converge to the most stable values and achieve the best convergence performance because of its collaboration feature.

Fig. 7(a) shows the computation cost of the above four training modes under different number of iterations. The computation cost is measured by the consumed time for training. By contrast, the centralized training mode needs the most training time because it requires the network controller to collect metrics data from all VM monitors for the centralized processing. The standalone training



Fig. 5. The diagram of different training modes.



Fig. 6. Training processes of BiWGAN-GP model in different training modes. (a) Centralized training mode; (b) Standalone training mode; (c) Multidiscriminator BiWGAN-GP model in distributed training mode; (d) Multi-discriminator BiWGAN-GP model in FL-based distributed training mode.

mode needs the least training time because each VM monitor trains the BiWGAN-GP model individually using its own local metrics data. Besides, the distributed and FL-based distributed training modes for multi-discriminator BiWGAN-GP model are more efficient than the centralized mode, but slightly inferior to the standalone mode.

In standalone mode, there is no communication among VM monitors, network slice managers and the network controller. In centralized mode, VM monitors send their all metrics data to the corresponding network slice managers and assemble on the network controller at the beginning. As the distributed and FL-based distributed training modes decentralize the discriminator training tasks to VM monitors by leaving generator and encoder training tasks on network slice managers, the communication cost between network slice managers and VM monitors is reasonable compared with the centralized mode. Fig. 7(b) shows the communication cost between network slice managers and VM monitors with the variation of batch sizes. The larger batch size brings a higher communication cost due to the increased data exchanged between network slice managers and VM monitors. Fig. 7(c) shows the communication cost between network slice managers and the network controller when increasing the number of local iterations L before aggregation. The communication cost decreases with the increase of L because the bigger L means less interactions between network slice managers and the network controller.

Fig. 7(d) shows the memory cost for the above four

training modes as the size of total training dataset increases. It shows that the memory cost on the network controller under distributed and FL-based distributed training modes is much less than that under the centralized training mode. Besides, it also indicates that the memory cost on the VM monitors under distributed and FL-based distributed training modes is also slightly less than that under the standalone training mode.

5.3 Effectiveness

After the FL-based multi-discriminator BiWGAN-GP model is trained well, the abnormality of new metrics data can be measured through its anomaly score calculated by Eq. (28). If the anomaly score is larger than a preset *threshold*, new metrics data will be determined as abnormal. The value of *threshold* will affect the detection performance of the FL-based multi-discriminator BiWGAN-GP model. The relationship between precision, recall and F1-score under different values of *threshold* is presented in Fig. 8, where each point is obtained by setting a new value of *threshold*.

From Fig. 8, we can see that different *threshold* values bring different detection performance, so we design a simple method to obtain an effective *threshold* value using the validation dataset. To simulate anomalies in VMs, four error types are injected to the validation dataset, including endless loop in CPU, memory leak, Disk I/O fault and network congestion. Then, we calculate the average



Fig. 7. Efficiency of different training modes. (a) Computation cost; (b) Communication cost between network slice managers (NSM) and VM monitors; (c) Communication cost between NSM and the network controller; (d) Memory cost.



Fig. 8. The relationship between precision, recall and F1-score.



Fig. 9. The detection performance comparison between BiWGAN-GP and the state-of-the-art GAN algorithms.

anomaly scores A_{normal} and $A_{abnormal}$ of normal and abnormal metrics in the modified validation dataset. The *threshold* is set as

$$threshold = \frac{A_{normal} + A_{abnormal}}{2}.$$
 (33)

BiWGAN-GP is the basic model used in the developed FL-based three-tier distributed anomaly detection framework. To validate the detection performance of the BiWGAN-GP algorithm and the state-of-the-art GAN algorithms in the presence of anomalies, four error types are injected to the test dataset randomly to simulate the anomalies in resource metrics data, including endless loop in CPU, memory leak, Disk I/O fault and network congestion. Fig. 9 shows the detection performance comparison between the BiWGAN-GP algorithm and the state-of-the-art GAN algorithms, including GAN [20], WGAN [21], WGAN-GP [23] and BiGAN [22]. For convenience, the standalone training mode is used for all the five algorithms in this simulation. Because the BiWGAN-GP algorithm combines both the strengths of BiGAN and WGAN-GP, it has obviously improved the detection performance compared to the state-of-the-art GAN algorithms. Specifically, compared to GAN, WGAN, WGAN-GP and BiGAN, the precision of BiWGAN-GP algorithm has increased by 4.5%, 2.9%, 2.2%, and 1.8%, respectively, the recall of BiWGAN-GP algorithm has increased by 4.8%, 2.2%, 2.1%, and 2.7%, respectively, and the F1-score of BiWGAN-GP algorithm has increased by 4.6%, 2.5%, 2.0%, and 2.2%, respectively.

Fig. 10 shows the detection performance of the centralized BiWGAN-GP, the standalone BiWGAN-GP, the multi-discriminator BiWGAN-GP and the FL-based multidiscriminator BiWGAN-GP algorithms. We use the precision, recall and F1-socre as the evaluation metrics. We can see that the values of three metrics using any algorithm are nearly stable after 300 iterations, which are consistent with the convergence trends of discriminator and EG losses presented in Fig. 6. It indicates that the four algorithms converge through enough iterations and can capture the distribution from metrics data of VMs. As shown in Fig. 10, the FL-based multi-discriminator BiWGAN-GP algorithm is superior to the standalone BiWGAN-GP and multidiscriminator BiWGAN-GP algorithms, and slightly inferior to the centralized BiWGAN-GP algorithm. Besides, according to the variation ranges of recall, precision and F1-score after 300 iterations, the detection performance of the FL-based multi-discriminator BiWGAN-GP algorithm is more stable than that of the standalone BiWGAN-GP and multi-discriminator BiWGAN-GP algorithms because of its collaboration feature.

The detection performance of four anomaly detection algorithms on four anomaly cases is summarized in Tables 2-5. Four anomaly detection algorithms are the centralized BiWGAN-GP, the standalone BiWGAN-GP, the multi-discriminator BiWGAN-GP and the FL-based multidiscriminator BiWGAN-GP algorithms. Four anomaly cases include endless loop in CPU, memory leak, Disk I/O fault and network congestion. By contrast, the performance of the FL-based multi-discriminator BiWGAN-GP algorithm is superior to the standalone BiWGAN-GP and multidiscriminator BiWGAN-GP algorithms, and slightly inferior to the centralized BiWGAN-GP algorithm on all four



Fig. 10. The detection performance comparison among four algorithms. (a) Precision; (b) Recall; (c) F1-score.

TABLE 2 Detection performance on endless loop in CPU

Algorithms	Accuracy	Precision	Recall	F1-score
Centralized	0.9842	0.9833	0.9848	0.9841
Standalone	0.9554	0.9549	0.9549	0.9549
Multi-discriminator	0.9651	0.9650	0.9645	0.9648
FL-based multi-discriminator	0.9740	0.9710	0.9774	0.9742

TABLE 3 Detection performance on memory leak

Algorithms	Accuracy	Precision	Recall	F1-score
Centralized	0.9860	0.9769	0.9892	0.9853
Standalone	0.9557	0.9563	0.9590	0.9560
Multi-discriminator	0.9673	0.9664	0.9683	0.9674
FL-based multi-discriminator	0.9743	0.9702	0.9783	0.9742

TABLE 4 Detection performance on Disk I/O fault

Algorithms	Accuracy	Precision	Recall	F1-score
Centralized	0.9874	0.9835	0.9916	0.9873
Standalone	0.9559	0.9544	0.9570	0.9557
Multi-discriminator	0.9667	0.9654	0.9672	0.9662
FL-based multi-discriminator	0.9740	0.9718	0.9754	0.9736

TABLE 5 Detection performance on network congestion

Algorithms	Accuracy	Precision	Recall	F1-score
Centralized	0.9726	0.9713	0.9735	0.9726
Standalone	0.9540	0.9501	0.9582	0.9542
Multi-discriminator	0.9635	0.9609	0.9659	0.9634
FL-based multi-discriminator	0.9740	0.9698	0.9754	0.9736

6 RELATED WORK

A substantial part of the paper lies at the intersection of the three following research domains: 1) Monitoring framework for network virtualization environment; 2) GANbased anomaly detection algorithm; 3) FL-based distributed learning paradigm.

6.1 Monitoring framework for network virtualization environment

Network virtualization environment can reduce the deployment cost and simplify the life-cycle management of network functions, but it introduces new management challenges and issues [38]. Therefore, building the effective monitoring framework for network virtualization environment is vital to ensure its high availability and reliability. To satisfy the security requirements of VMs, a comprehensive protection mechanism with the capacity of defense-in-depth for VMs was proposed in [39], where the anomaly detection was realized by building the normal traffic model and determining the abnormal behaviors according to their degree of deviation from the normal behaviors. Mishra et al. [40] designed a multi-level security architecture, namely VMGuard, for VM monitoring at the process level and system call level to detect known attacks and their variants. VMGuard applied the random forest classifier to produce a generic behavior for different categories of attacks in monitored VMs. As container-based virtualization has become a key solution in present virtualized networks, Zou et al. [4] developed an online container anomaly detection framework by monitoring and analyzing the resourcerelevant metrics of containers with the optimized isolation forest algorithm. The considered metrics included CPU usage, memory usage, disk read/write speed and network speed. However, the monitoring framework designed in [4], [39], [40] used centralized database to collect and process relevant data of VMs distributed in different regions of substrate networks, which will cause high communication and computation overhead when learning a global VM anomaly detection model. Our previous work [41] designed a distributed monitoring mechanism from the perspective of physical nodes and links, not from the entire networks. In this paper, for each region of the entire networks, we deploy a VM monitor for a network slice to collect and store metrics

data of VMs in the network slice as a local database. The distributed anomaly detection can be trained over datasets that are spread across the entire networks.

6.2 GAN-based anomaly detection algorithm

GAN is a promising generative model proposed by Goodfellow et al. [33], which is superior in capturing the distribution from high-dimensional complex real-world data. GAN consists of two models: the generator and the discriminator, which are usually built with neural networks. By training the generator and discriminator through the adversarial learning, the Nash equilibrium can be achieved theoretically [42], where the generator can generate fake samples sharing the same distribution with real ones that the discriminator is unable to distinguish between them. From this, GAN and its improved modifications have been widely used to implement anomaly detection in various fields, such as in distribution systems [14], [43], in vehicular ad hoc networks [44], in high-speed railways [45] and in movie reviews [37]. The rationale behind the GAN-based anomaly detection is capturing the distribution from normal data and then finding an effective discriminant criterion to distinguish abnormal data from normal ones. In [14], BiWGAN was used as a feature extractor to map normal samples back to latent space and support vector data description was used to establish the discriminant criterion for nontechnical losses detection in power system. Using smart meters' data, a GAN was designed in [43] to extract the normal temporalspatial behavior of distribution systems. When abnormal behaviors occurred, the smart meters' data were expected to have anomaly scores out of the normal range. Shu et al. [44] proposed a distributed BiGAN framework to detect abnormal network behaviors for the entire vehicular ad hoc networks without exchanging data belonging to local sub-networks. In [45], a deep convolutional GAN was constructed to map the images of catenary support components into high-dimensional feature spaces, and an anomaly rating criterion was defined to detect abnormal images of high-speed railways. Gao et al. [37] developed a novel attention-driven conditional GAN to capture the correlations of movie reviews and identify spammed reviews. The effectiveness of GAN and its improved modifications in anomaly detection has been fully verified through extensive researches in these fields. In this paper, we design a new multi-discriminator BiWGAN-GP algorithm with multiple discriminators running locally in view of the distributed data features in virtualized network slicing environment.

6.3 FL-based distributed learning paradigm

FL is developed as a distributed leaning framework where training data are decentralized across learners, rather than being centralized [46], [47], [48], [49]. FL allows each learner to train a model from local data and send its local model to a center periodically. The center is responsible for model aggregation and sharing the global model with all learners. In [46], to model reliability in vehicular communications with low communication cost, FL was used to enable vehicular users to learn the tail distribution of network-wide queue lengths locally without sharing the actual queue data. As anomalies in edge devices seriously affect the manufacturing process in industrial IoT, a communication-efficient FL based deep anomaly detection framework was proposed in [47] to enable distributed edge devices to collaboratively train a global detection model with improved generalization ability. Darwish et al. [48] extended FL to achieve the self-evolving network management for future intelligent vertical HetNet. In [49], based on the three-layer association relationship in radio access network slicing (device-base station-network slicing), a hybrid FL reinforcement learning framework was exploited to solve the device association problem while enforcing data security and device privacy, which included a horizontal aggregation on base stations for the same service types and a vertical aggregation on an encrypted center for different service types. Inspired by the existing researches, FL-based distributed learning paradigm can well adapt to the hierarchical control of virtualized network slicing, which facilitates the establishment of a global anomaly detection model without compromising the communication and computation efficiency.

7 CONCLUSIONS

The novelty of the paper lies in the combination of multidiscriminator BiWGAN-GP and FL to detect anomalies in VMs in the context of virtualized network slicing environment. Within a network slice, the multi-discriminator BiWGAN-GP algorithm implements discriminators on VM monitors, and the generator and encoder on its network slice manager to monitor and analyze the resource metrics data of VMs in a distributed way. Then, the updated parameters of generators and encoders trained on network slice managers are sent to the network controller for aggregation using FL framework. We achieve a global VM anomaly detection model through the hierarchical cooperation among VM monitors, network slice managers and the network controller. We define an anomaly score using the reconstruction loss and discriminator loss of the multidiscriminator BiWGAN-GP algorithm, and then adopt the validation dataset to obtain the effective threshold for the anomaly score to distinguish between normal and abnormal behaviors. The experiment results on a real-world dataset validate the efficiency and effectiveness of the proposed FL-based multi-discriminator BiWGAN-GP algorithm on detecting typical anomalies of VMs in virtualized network slicing environment.

REFERENCES

- J. Ordonez-Lucena *et al.*, "Network slicing for 5G with SDN/NFV: Concepts, architectures, and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 80–87, May 2017.
- [2] J. Khamse-Ashari, G. Senarath, I. Bor-Yaliniz, and H. Yanikomeroglu, "An agile and distributed mechanism for inter-domain network slicing in next-generation mobile networks," *IEEE Trans. Mobile Comput.*, early access, Feb. 23, 2021, doi: 10.1109/TMC.2021.3061613.
- [3] F. Z. Yousaf, M. Bredel, S. Schaller, and F. Schneider, "NFV and SDN-Key technology enablers for 5G networks," *IEEE J. Sel. Areas Commun.*, vol. 35, no. 11, pp. 2468–2478, Nov. 2017.
- [4] Z. Zou *et al.*, "A docker container anomaly monitoring system based on optimized isolation forest," *IEEE Trans. Cloud Comput.*, vol. 10, no. 1, pp. 134–145, Jan.-Mar. 2022.

- [5] H. Zhang, J. Liu, and T. Wu, "Adaptive and incremental-clustering anomaly detection algorithm for VMs under cloud platform runtime environment," *IEEE Access*, vol. 6, pp. 76984–76992, Dec. 2018.
- [6] R. Cogranne, G. Doyen, N. Ghadban, and B. Hammi, "Detecting botclouds at large scale: A decentralized and robust detection method for multi-tenant virtualized environments," *IEEE Trans. Netw. Service Manag.*, vol. 15, no. 1, pp. 68–82, Mar. 2018.
- [7] C. Sauvanaud *et al.*, "Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned," J. Syst. Softw., vol. 139, pp. 84–106, May 2018.
- [8] H. Liu *et al.*, "Rain: Towards real-time core devices anomaly detection through session data in cloud network," in *Proc. NOMS-IEEE/IFIP Netw. Operations Manage. Symp.*, Apr. 2020, pp. 1–6.
- [9] W. Wang, Q. Chen, X. He, and L. Tang, "Cooperative anomaly detection with transfer learning-based hidden markov model in virtualized network slicing," *IEEE Commun. Lett.*, vol. 23, no. 9, pp. 1534–1537, Sep. 2019.
- [10] Q. Jiang and X. Yan, "Multimode process monitoring using variational bayesian inference and canonical correlation analysis," *IEEE Trans. Autom. Sci. Eng.*, vol. 16, no. 4, pp. 1814–1824, Oct. 2019.
- [11] O. Onireti *et al.*, "A cell outage management framework for dense heterogeneous networks," *IEEE Trans. Veh. Technol.*, vol. 65, no. 4, pp. 2097–2113, Apr. 2016.
- [12] Q. Liao and S. Stanczak, "Network state awareness and proactive anomaly detection in self-organizing networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2015, pp. 1–6.
- [13] X. Miao, Y. Liu, H. Zhao, and C. Li, "Distributed online one-class support vector machine for anomaly detection over networks," *IEEE Trans. Cybern.*, vol. 49, no. 4, pp. 1475–1488, Apr. 2019.
- [14] T. Hu et al., "Nontechnical losses detection through coordinated BiWGAN and SVDD," IEEE Trans. Neural Netw. Learn. Syst., vol. 32, no. 5, pp. 1866–1880, May 2021.
- [15] J. Chen, M. Chen, X. Wei, and B. Chen, "Matrix differential decomposition-based anomaly detection and localization in NFV networks," *IEEE Access*, vol. 7, pp. 29 320–29 331, Jan. 2019.
- [16] Y. Cheng, H. Zhu, J. Wu, and X. Shao, "Machine health monitoring using adaptive kernel spectral clustering and deep long shortterm memory recurrent neural networks," *IEEE Trans. Ind. Inform.*, vol. 15, no. 2, pp. 987–997, Feb. 2019.
- [17] Z. Li, A. L. G. Rios, and L. Trajković, "Machine learning for detecting anomalies and intrusions in communication networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 7, pp. 2254–2264, Jul. 2021.
- [18] I. Nevat *et al.*, "Anomaly detection and attribution in networks with temporally correlated traffic," *IEEE/ACM Trans. Netw.*, vol. 26, no. 1, pp. 131–144, Feb. 2018.
- [19] M. A. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Process.*, vol. 99, no. 6, pp. 215–249, Jun. 2014.
- [20] D. Giannopoulos, P. Papaioannou, C. Tranoris, and S. Denazis, "Monitoring as a service over a 5G network slice," in *Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit)*, Jun. 2021, pp. 329–334.
- [21] V. A. Cunha et al., "5 Growth: Secure and reliable network slicing for verticals," in Proc. Joint Eur. Conf. Netw. Commun. 6G Summit (EuCNC/6G Summit), Jun. 2021, pp. 347–352.
- [22] C. Papagianni et al., "5Growth: AI-driven 5G for automation in vertical industries," in Proc. Eur. Conf. Netw. Commun. (EuCNC), Jun. 2020, pp. 17–22.
- [23] A. Creswell *et al.*, "Generative adversarial networks: An overview," *IEEE Signal Process. Mag.*, vol. 35, no. 1, pp. 53–65, Jan. 2018.
- [24] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Dec. 2017, arXiv:1701.07875. [Online]. Available: http://arxiv.org/abs/ 1701.07875.
- [25] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," Apr. 2017, arXiv:1605.09782. [Online]. Available: http://arxiv.org/abs/1605.09782.
- [26] I. Gulrajani *et al.*, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, May 2017, pp. 5769–5779.
 [27] Y. Liu *et al.*, "Privacy-preserving traffic flow prediction: A
- [27] Y. Liu *et al.*, "Privacy-preserving traffic flow prediction: A federated learning approach," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7751–7763, Aug. 2020.
 [28] J. Koneŏný *et al.*, "Federated learning: Strategies for improving
- [28] J. Koneŏný et al., "Federated learning: Strategies for improving communication efficiency," Oct. 2017, arXiv:1610.05492. [Online]. Available: https://arxiv.org/abs/1610.05492.

- [29] "OSM: Open-Source MANO," 2016. [Online]. Available: https: //osm.etsi.org/.
- [30] "OpenStack Tacker," 2017. [Online]. Available: https://wiki. openstack.org/wiki/Tacker.
- [31] A. Laghrissi and T. Taleb, "A survey on the placement of virtual resources and virtual network functions," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1409–1434, 2nd Quart. 2019.
- [32] "OpenBaton," 2015. [Online]. Available: https://openbaton. github.io/.
- [33] I. J. Goodfellow et al., "Generative adversarial nets," in Proc. Adv. Neural Inf. Process. Syst., Jun. 2014, pp. 2672–2680.
- [34] K. Greff et al., "LSTM: A search space odyssey," IEEE Trans. Neural Netw. Learn. Syst., vol. 28, no. 10, pp. 2222–2232, Oct. 2017.
- [35] C. Hardy, E. Le Merrer, and B. Sericola, "MD-GAN: Multidiscriminator generative adversarial networks for distributed datasets," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, May 2019, pp. 866–877.
- [36] T. Schlegl et al., "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in Proc. Int. Conf. Inf. Process. Med. Imag., May 2017, pp. 146–157.
- [37] Y. Gao, M. Gong, Y. Xie, and A. K. Qin, "An attentionbased unsupervised adversarial model for movie review spam detection," *IEEE Trans. Multimedia*, vol. 23, pp. 784–796, Jan. 2021.
- [38] S. Cherrared et al., "A survey of fault management in network virtualization environments: Challenges and solutions," IEEE Trans. Netw. Service Manag., vol. 16, no. 4, pp. 1537–1551, Dec. 2019.
- [39] X. Yin et al., "Research of security as a service for VMs in IaaS platform," IEEE Access, vol. 6, pp. 29158–29172, Jun. 2018.
- [40] P. Mishra, V. Varadharajan, E. S. Pilli, and U. Tupakula, "VMGuard: A VMI-based security architecture for intrusion detection in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 8, no. 3, pp. 957–971, Jul.-Sep. 2020.
 [41] W. Wang *et al.*, "Distributed online anomaly detection for virtual-
- [41] W. Wang *et al.*, "Distributed online anomaly detection for virtualized network slicing environment," *IEEE Trans. Veh. Technol.*, pp. 1–15, early access, Jul. 2022, doi: 10.1109/TVT.2022.3193074.
- [42] K. Wang et al., "Generative adversarial networks: Introduction and outlook," IEEE/CAA J. Autom. Sinica, vol. 4, no. 4, pp. 588–598, Sep. 2017.
- [43] Y. Yuan, K. Dehghanpour, F. Bu, and Z. Wang, "Outage detection in partially observable distribution systems using smart meters and generative adversarial networks," *IEEE Trans. Smart Grid*, vol. 11, no. 6, pp. 5418–5430, Nov. 2020.
- [44] J. Shu et al., "Collaborative intrusion detection for VANETs: A deep learning-based distributed SDN approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 7, pp. 4519–4530, Jul. 2021.
- [45] Y. Lyu et al., "A generic anomaly detection of catenary support components based on generative adversarial networks," IEEE Trans. Instrum. Meas., vol. 69, no. 5, pp. 2439–2448, May 2020.
- [46] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Feb. 2020.
- [47] Y. Liu *et al.*, "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348– 6358, Apr. 2021.
- [48] T. Darwish *et al.*, "A vision of self-evolving network management for future intelligent vertical hetnet," *IEEE Wireless Commun.*, vol. 28, no. 4, pp. 96–105, Aug. 2021.
 [49] Y.-J. Liu *et al.*, "Device association for RAN slicing based on hybrid
- [49] Y.-J. Liu *et al.*, "Device association for RAN slicing based on hybrid federated deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 69, no. 12, pp. 15731–15745, Dec. 2020.