WOOD: Wasserstein-Based Out-of-Distribution Detection

Yinan Wang, Member, IEEE, Wenbo Sun, Jionghua Jin, Zhenyu Kong, Member, IEEE, and Xiaowei Yue, Senior Member, IEEE

Abstract—The training and testing data for deep-neural-network-based classifiers are usually assumed to be sampled from the same distribution. When part of the testing samples are drawn from a distribution that is sufficiently far away from that of the training samples (a.k.a. out-of-distribution (OOD) samples), the trained neural network has a tendency to make high-confidence predictions for these OOD samples. Detection of the OOD samples is critical when training a neural network used for image classification, object detection, etc. It can enhance the classifier’s robustness to irrelevant inputs, and improve the system’s resilience and security under different forms of attacks. Detection of OOD samples has three main challenges: (i) the proposed OOD detection method should be compatible with various architectures of classifiers (e.g., DenseNet, ResNet) without significantly increasing the model complexity and requirements on computational resources; (ii) the OOD samples may come from multiple distributions, whose class labels are commonly unavailable; (iii) a score function needs to be defined to effectively separate OOD samples from in-distribution (InD) samples. To overcome these challenges, we propose a Wasserstein-based out-of-distribution detection (WOOD) method. The basic idea is to define a Wasserstein-based score that evaluates the dissimilarity between a test sample and the distribution of InD samples. An optimization problem is then formulated and solved based on the proposed score function. The statistical learning bound of the proposed method is investigated to guarantee that the loss value achieved by the empirical optimizer approximates the global optimum. The comparison study results demonstrate that the proposed WOOD consistently outperforms other existing OOD detection methods.

Index Terms—Cyber security, image classification, machine learning, OOD detection, Wasserstein distance.

I. INTRODUCTION

Deep Neural Networks (DNNs) have achieved outstanding performances on many challenging tasks, such as image classification [1], [2], object detection [3], [4], and speech recognition [5]. When training a DNN, it is often assumed that the training and testing samples are drawn from the same distribution. However, in practice, there likely exist abnormal testing samples that are drawn from other distributions. Compared to the samples drawn from the distribution of training samples (a.k.a. in-distribution (InD) data), these abnormal samples are referred to as out-of-distribution (OOD) data, which may not belong to any of the classes that the model is trained on. In this situation, a DNN-based classifier tends to over-confidently predict OOD samples with the class labels of InD samples. Therefore, there is a pressing need to boost the existing DNN-based classifier with the capability for OOD detection while preserving its performance on the InD tasks.

OOD detection for a DNN-based classifier mainly includes two sub-tasks: (i) A score function needs to be properly defined to evaluate the difference between InD and OOD samples; (ii) The InD and OOD samples need to be further separated in the space defined by the score function. These two sub-tasks are coupled with each other. A well-defined score function can better capture the difference and naturally separate InD and OOD samples, and such differences can be further enlarged by various techniques, such as preprocessing the input, calibrating the hyperparameters used in the score function, and retraining the classifier.

In this paper, we propose the Wasserstein-based OOD detection (WOOD) method, which is a unified framework to train a DNN-based classifier to conduct classification and OOD detection simultaneously. The “Wasserstein” here refers to the Wasserstein distance [6], which is designed to evaluate the difference between probability distributions. In the following section, we first categorize and review some recent studies in OOD detection and then discuss the necessity and current research gaps in adapting the Wasserstein distance as a dissimilarity measure and a loss function for OOD detection.

A. Related Work

1) OOD Detection Without Tuning the Pre-Trained Classifier: The recently developed methods of OOD detection in classification can be summarized into two categories according to whether the classifier is re-trained on the OOD samples. In

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
the first category, OOD detection is regarded as an augmented property for a pre-trained classifier. For example, Hendrycks and Gimpel built a baseline for OOD detection using the threshold-based detector and the maximum softmax score as the score function [7]. Liang et al. further rescaled the softmax score with a temperature parameter and preprocessed the inputs to further enlarge the difference between the maximum softmax score of InD and OOD samples [8]. Rather than relying on the maximum softmax score, some researchers tried to define the score functions based on different distance measures. For example, Hsu et al. improved Liang’s method by decomposing the confidence score and modified the input processing method [9]. The Mahalanobis distance was calculated and calibrated on the intermediate features of DNNs to serve as the confidence score [10], [11]; A measure of confidence was proposed by analyzing the invariance of softmax score under various transformations of inputs [12]; The uncertainty of DNNs was evaluated by using the gradient information from all the layers to serve as the score function [13]; The trust score for each input was defined as the ratio of the Hausdorff distances from the input to its closest and second closest labels, which is used to determine whether a classifier’s prediction can be trusted or not [14]; Zaeemzadeh et al. argued that OOD samples could be better detected in a lower-dimension embedding space. They proposed to reduce the embedding space of the existing classifier and further applied cosine similarity between first singular vectors as the score function to detect OOD samples [15]. By projecting the inputs into a new space, these newly defined score functions can distinguish the InD and OOD samples better than the methods relying on the softmax score.

Although these methods are compatible with different neural network architectures and are easy to implement, the score-function-based OOD detection methods have two limitations: (i) The choice of score functions depends on the type of applications and OOD patterns. It is required to design a score function that has a consistently good performance under all OOD patterns. (ii) The OOD detection performance highly relies on the values of hyperparameters in the selected score function, which are often tuned based on auxiliary OOD samples. However, after the classifier has been pre-trained, simply tuning the hyperparameters in the score function may not guarantee a good separation of InD and OOD samples, which may not achieve the desired OOD detection performance.

2) OOD Detection by Re-Training Classifiers: In the second category, researchers tried to propose a unified framework to simultaneously train an OOD detector during the training process of the neural-network-based classifier. For example, Hendrycks et al. [16] proposed an outlier exposure method to leverage auxiliary datasets of OOD outliers for training classifiers. An adversarial training with the informative outlier mining (ATOM) method was proposed to improve the robustness of OOD detection with auxiliary datasets [17]. Moreover, DeVries and Taylor [18] proposed to add a confidence branch on the top of a classifier and train this branch to generate binary outputs to represent whether one input sample is OOD or not. Mohseni et al. [19] added an auxiliary head on the top of a classifier and trained it in a self-supervised way. Liu et al. proposed to include an energy-based score function in the training process of the classifier to boost its performance on OOD detection with auxiliary datasets [20].

These methods can identify some fundamental differences between InD and OOD samples when training the classifier. However, they have three limitations: (i) The score function used in the training process is not properly designed to fully represent the difference between two discrete distributions accurately; (ii) Additional components added to a classifier will increase the model complexity and training time; (iii) Massive amount of OOD samples are often introduced during training. It is also unrealistic to assume all the OOD samples are from the same class. In practice, human beings usually categorize one sample as OOD when it has significant differences from InD samples and does not belong to any of the InD classes. By forcing different types of OOD samples to fall in the same class, this category of OOD detection methods brings additional difficulty in training the classification model, which may result in poor performances in OOD detection and InD classification.

3) Reviews of Distance Metrics for Probability Distribution: The softmax output from a classifier for a specific input is the discrete probability distribution over all the possible classes. Thus, the dissimilarity between output distributions directly reflects the difference between inputs. Wasserstein distance is often used to measure the dissimilarity between probability distributions. Compared with the Kullback-Leibler (KL) divergence and Jensen-Shannon (JS) divergence, Wasserstein distance provides a meaningful and smooth representation of the distance even when two distributions are located in lower-dimensional manifolds without overlaps [21]. The smooth measure of distance ensures a stable learning process when optimizing the Wasserstein distance with the gradient-based method. Wasserstein distance has demonstrated its strength in many applications, such as the distributionally robust stochastic optimization (DRSO) [22], deep active learning [23], reinforcement learning [24], image classification [25], and data augmentation [26].

In this paper, we use Wasserstein distance to measure the dissimilarity between the InD and OOD samples, which is further used to detect OOD samples. To the best of our knowledge, our proposed WOOD is the first work to design the OOD detection framework by leveraging the strength of Wasserstein distance in measuring the dissimilarity between distributions. Although there are prior works adapting Wasserstein distance into the loss function of a classification task [25] and designing the Generative and Adversarial Networks (GAN) [27], research gaps still exist in designing OOD detection framework with Wasserstein distance: (1) Unlike the classification task, the label information of OOD samples are commonly unavailable, which make the Wasserstein-based loss function in the classification task invalid to be used in OOD detection; (2) Instead of a single objective in previous works, the proposed WOOD framework is required to train the model to do classification and OOD detection simultaneously.
B. Motivations and Contributions

To overcome the above limitations in using Wasserstein distance for OOD detection, we propose a new integrative approach, named WOOD, which integrates the above two categories of OOD detection methods (Sections I-A1 and I-A2). Specifically, we design a new OOD score function whose hyperparameters can be simultaneously updated with the classifier. Fig. 1 shows a comparison of the classifier without OOD detection, the two categories of the existing OOD detection methods, and the proposed method. Going beyond the original neural-network-based classifier in Fig. 1(a) [1], [2], Fig. 1(b) demonstrates the flowchart of the first category of the existing methods, which maps inputs to a specific score function by preprocessing the inputs and postprocessing the outputs [7], [8], [9], [10], [11], [12], [13], [14], [15]. However, without simultaneously training the DNN for classification and OOD detection, it is often challenging to find a set of optimal hyperparameters to consistently identify various types of OOD samples. Fig. 1(c) shows the idea of simultaneously training the classifier with InD and auxiliary OOD samples [16], [17], [18], [19], [20]. The advantage of these methods is the efficiency of realizing sample classification and OOD detection simultaneously. However, they rely on the assumption that all the OOD samples belong to the same class, which may not hold when different types of OOD samples are included in the test dataset.

Motivated by the aforementioned research gaps, as shown in Fig. 1(d), we propose the Wasserstein-based score function and integrate it with the unified framework for OOD detection. The definition of our score function is intuitive. Suppose a classifier is trained to output the probabilities of a sample falling in each potential class. For an InD sample, we would like the corresponding prediction probabilities to be close to a one-hot vector, whose elements involve a close-to-one value for the true class and close-to-zero values for all the other classes. For an OOD sample, we would like the prediction probabilities to
be fuzzy among most classes, that is, a vector whose multiple elements take similar values.

In our proposed WOOD, Wasserstein distance [6] is adopted to quantitatively measure such a distance between output discrete probability distributions when building the OOD score function, which is further incorporated into designing the multi-objective loss function. Wasserstein distance has two advantages: (i) it is defined in the metric space to evaluate the dissimilarity between distributions, which makes it a natural fit to the OOD detector; (ii) it can be optimized with the gradient-based method, so it is resilient to and compatible with various designs of loss function and diverse classifier structures. Compared to the two categories of the existing OOD detection methods, the contributions of the proposed WOOD method are four folds:

1) A Wasserstein-based score function is proposed for the separation of InD and OOD samples. The definition of the score function is intuitive and will be shown to outperform the benchmark score functions in several commonly used datasets in the machine learning field.

2) A unified framework is designed to simultaneously train the classifier and OOD detector. Such a one-step optimization setup will not introduce additional model complexity or computational load.

3) The proposed OOD detection method does not require detailed labels of the OOD samples nor force all the OOD samples to belong to the same class. With a relaxed assumption on the OOD samples, the proposed method can be generally applied to various applications.

4) The theoretical properties of WOOD are investigated, which guarantees that the loss value achieved by the empirical minimizer approaches the global optimum. The gradient of the proposed loss function in WOOD is explicitly derived to further save computation time.

The remainder of this paper is organized as follows. Section II proposes the formulation of the WOOD method and discusses the properties and computational complexity of WOOD with two different cost matrices. Section III develops the learning algorithm of WOOD and derives the statistical learning bound of the proposed loss function. Section IV demonstrates the performance of WOOD using multiple datasets. Finally, Section V summarizes the contributions and concludes this paper.

II. PROPOSED WASSERSTEIN-BASED OUT-OF-DISTRIBUTION DETECTION

We show the technical details of the proposed Wasserstein-based out-of-distribution detection (WOOD) method in this section. First, we briefly introduce the problem setup. Then, we formally define the OOD score function, incorporate it into the loss function, and formulate the corresponding detector for OOD samples. Lastly, we formulate the training of the OOD detector as an optimization problem and describe the procedure to obtain the numerical solution.

A. Problem Setup

In a classification problem, let $\mathcal{X}$ denote the space of inputs. Let $\mathcal{X}_{\text{InD}}$ and $\mathcal{X}_{\text{OOD}}$ denote two distinct sub-spaces on $\mathcal{X}$. Specifically, $\mathcal{X}_{\text{InD}}$ represents the space of InD samples with labels from $\mathcal{K}_{\text{InD}}$, and $\mathcal{X}_{\text{OOD}}$ represents the space of OOD samples whose labels lie beyond $\mathcal{K}_{\text{InD}}$. Here $\mathcal{X} = \mathcal{X}_{\text{InD}} \cup \mathcal{X}_{\text{OOD}}$. We refer to the input data from $\mathcal{X}_{\text{InD}}$ and $\mathcal{X}_{\text{OOD}}$ as In-distribution (InD) samples and Out-Of-Distribution (OOD) samples, respectively. Our objective is to train a classifier to correctly label the InD inputs from $\mathcal{X}_{\text{InD}}$, and a detector to identify samples from $\mathcal{X}_{\text{OOD}}$.

Our training dataset is organized as follows. Let $D_{\text{InD}}$ denote the training dataset of InD samples. Each sample in $D_{\text{InD}}$ consists of an input from $\mathcal{X}_{\text{InD}}$, and a corresponding class label from $\mathcal{K}_{\text{InD}}$. Since it is difficult to train an accurate OOD detector based on InD samples only [17], we consider the case where the training dataset of auxiliary OOD samples, $D_{\text{OOD}}$, is available. In real practice, $D_{\text{OOD}}$ can be generated from adversarial training techniques [28] when real OOD samples are difficult to collect. Each element in the OOD dataset is an unlabelled input from $\mathcal{X}_{\text{OOD}}$ because OOD samples may come from various classes without exact labels.

Given the datasets described above, our objective is two-fold. First, for an InD sample $x_{\text{InD}} \in \mathcal{X}_{\text{InD}}$ with label $k \in \mathcal{K}_{\text{InD}}$, we would like the predicted probability by the classifier to satisfy the following condition:

$$f_0(x_{\text{InD}}) \approx y^k,$$

where $y^k$ is an one-hot vector whose $k$th element is 1 and other elements are 0’s; $f_0$ is a classifier that is parametrized by the model parameter $\theta$ and outputs the probabilities of an input $x_{\text{InD}}$ falling into all the class in $\mathcal{K}_{\text{InD}}$. Second, for an arbitrary sample $x \in \mathcal{X}$, we would like to train an OOD detector $g$ such that

$$g(x) = \begin{cases} 1, & \text{if } x \in \mathcal{X}_{\text{OOD}} \\ 0, & \text{if } x \notin \mathcal{X}_{\text{InD}}. \end{cases}$$

The true negative rate (TNR) and false negative rate (FNR) defined on the OOD detector $g$ are

$$\text{TNR}(g) = \mathbb{E}_{x \in \mathcal{X}_{\text{InD}}} \mathbb{I}[g(x) = 0],$$

$$\text{FNR}(g) = \mathbb{E}_{x \in \mathcal{X}_{\text{OOD}}} \mathbb{I}[g(x) = 0],$$

where $\mathbb{I}[\cdot]$ is a sign function, when $g(x) = 0$, its value is 1, otherwise, its value is 0. TNR denotes the percentage of InD samples that are correctly identified, and FNR is the percentage of OOD samples that are misidentified as InD samples.

B. Wasserstein-Based Score Function

We start from the general formulation of the Wasserstein distance for discrete distributions. Suppose $r_1$ and $r_2$ are discrete distributions of random variables that take values in $K$ classes with labels $\{1, \ldots, K\}$, the Wasserstein distance between two $r_1$ and $r_2$ [25] is

$$W(r_1, r_2) = \inf_{P \in \Pi(r_1, r_2)} \langle P, M \rangle,$$

where $M \in \mathbb{R}^{K \times K}_+$ is the cost matrix; $\langle \cdot, \cdot \rangle$ represents Kronecker product between two matrices; $P$ is the joint probability distribution with an element of

$$\Pi(r_1, r_2) := \left\{ P \in \mathbb{R}^{K \times K}_+ \mid P1_K = r_2, P^\top 1_K = r_1 \right\},$$

where $1_K \in \mathbb{R}^K$ denotes the $K$-dimensional vector whose elements are all ones; $r_1$ and $r_2$ are the marginal distributions of $P$. The remainder of this paper is organized as follows. Section II proposes the formulation of the WOOD method and discusses the properties and computational complexity of WOOD with two different cost matrices. Section III develops the learning algorithm of WOOD and derives the statistical learning bound of the proposed loss function. Section IV demonstrates the performance of WOOD using multiple datasets. Finally, Section V summarizes the contributions and concludes this paper.
The properties of Wasserstein distance are included in Appendix A, available online.

As was mentioned in Section I, an OOD score function should indicate how likely an incoming sample belongs to $\mathcal{X}_{\text{OOD}}$. The Wasserstein distance is adapted to design the OOD score function by leveraging its advantage in evaluating the dissimilarity between distributions. Let $K$ denote the number of different classes in $\mathcal{K}_{\text{Ind}}$. For $x \in \mathcal{X}$, the Wasserstein distance between the predictions given by $f_\theta$ and the one-hot vector $y^k$ of class $k$ is computed as:

$$W(f_\theta(x), y^k) = \inf_{P} \langle P, M \rangle,$$

s.t. $P 1_K = y^k$,

$$P^T 1_K = f_\theta(x),$$

$$k \in \mathcal{K}_{\text{Ind}},$$

(6)

where $P$ is a joint discrete probability distribution whose marginal distributions are $f_\theta(x)$ and $y^k$. It is worth noting that the feasible region of $P$ might change when calculating Wasserstein distances between different pairs of marginal distributions. $M$ is the distance matrix indicating the cost to transport one unit mass of probability between two discrete distributions. Its structure will be specified in Section II-D. For an OOD sample $x_{\text{OOD}} \in \mathcal{X}_{\text{OOD}}$, we would like the distance $W(f_\theta(x_{\text{OOD}}), y^k)$ to be large for each class $k \in \mathcal{K}_{\text{Ind}}$, which is equivalent to enlarge the distance to its closest class. Thus, for a random sample $x$, the proposed Wasserstein-based score function is defined as:

$$s(x) = \min_{k} W(f_\theta(x), y^k)$$

$$= \min_{k} \inf_{P} \langle P, M \rangle,$$

s.t. $P 1_K = y^k$,

$$P^T 1_K = f_\theta(x),$$

$$k \in \mathcal{K}_{\text{Ind}}.$$  

(7)

Note that the above Wasserstein-based score function does not require the assumption that the OOD samples belong to the same class. Instead, it reveals the difference between InD and OOD samples by evaluating the closest distance from an arbitrary sample to any InD classes. Ideally, if $x$ is an InD sample, $s(x)$ should be close to 0. If $x$ is an OOD sample, $s(x)$ should be close to a specific positive value, which is determined by the choice of the distance matrix $M$. To this end, we impose a threshold parameter $\epsilon$ on the defined Wasserstein-based score function and construct the OOD detector $g$ as:

$$g(x) = \begin{cases} 
1, & s(x) > \epsilon \\
0, & s(x) \leq \epsilon,
\end{cases}$$

(8)

The training of the threshold parameter $\epsilon$ will be discussed in Section IV-A2.

C. WOOD Loss Function

With the specifically designed score function, the instant question is how to train the classifier to separate InD and OOD samples in the space defined by the score function, as well as preserve the classification performance. As a key component for training the classifier, a new loss function is designed for OOD detection by mimicking the human intuitive logic, that is, to regard a sample as OOD if it does not belong to any existing classes based on human experience/memory. The Wasserstein-based score function is incorporated in the loss, and the classifier is trained to assign the InD samples with their correct labels and keep all the OOD samples away from any InD classes, as shown in Fig. 2. Motivated by this logic, we define the WOOD loss function as

$$L(\mathcal{D}_{\text{Ind}}, \mathcal{D}_{\text{OOD}}) = \frac{1}{N_{\text{Ind}}} \sum_{(x_{\text{Ind}}, k) \in \mathcal{D}_{\text{Ind}}} - \log \left( f_\theta(x_{\text{Ind}})^T y^k \right)$$

$$- \beta \frac{1}{N_{\text{OOD}}} \sum_{x_{\text{OOD}} \in \mathcal{D}_{\text{OOD}}} \min_{k} \inf_{P} \langle P, M \rangle,$$

s.t. $P 1_K = y^k$,

$$P^T 1_K = f_\theta(x_{\text{OOD}}),$$

$$k \in \mathcal{K}_{\text{Ind}},$$

(9)

where $N_{\text{Ind}}, N_{\text{OOD}}$ are the sample sizes of $\mathcal{D}_{\text{Ind}}, \mathcal{D}_{\text{OOD}}$, respectively; $\beta$ is the hyperparameter to balance the loss in the classification error and OOD detection error. The WOOD loss includes two terms. The first term on the right-hand side of (9) is the cross-entropy loss that aims to correctly match the InD samples with the corresponding labels. The second term on the right-hand side of (9) is the Wasserstein-based score, which calculates the minimum distances between the predicted softmax score vector and the one-hot vector of any InD classes.

By minimizing the WOOD loss, the classifier will tend to assign correct labels to InD samples and keep the predicted results of OOD samples away from any of the InD classes.

D. Properties of WOOD With Different Distance Matrices

The properties of WOOD are influenced by the choices of distance matrices $M$ in (9). Here we provide two commonly used distance matrices - the binary distance matrix and dynamic distance matrix and discuss their influence on the properties of WOOD.

1) Binary Distance Matrix: The formulation of the binary distance matrix is given in (10). Recall that the distance matrix in the Wasserstein distance is to evaluate the unit cost of transporting the probability mass. In the binary distance matrix $M_B$, the diagonal entries are set as 0, while the other entries are set as 1. Given that the binary distance matrix treats all the classes equally, transporting the unit probability mass to any different class will consistently cause a cost of 1.

$$M_B = \begin{bmatrix} 
0 & 1 & \ldots \\
\vdots & \ddots & \ddots \\
1 & \ldots & 0
\end{bmatrix}$$

(10)

The binary distance matrix is widely used in applications of Wasserstein distance, such as classification [25]. Our work is the first to explore its performance in OOD detection. The
advantages of binary distance matrix include: (i) the formulation is straightforward and easy to understand; (ii) it is compatible with various classifier structures and diverse applications; (iii) it is a metric matrix so that the Wasserstein distance satisfies the axioms of a distance (shown in (22), which is in Appendix A and available online). In addition, the maximum point of the Wasserstein-based score function with the binary distance matrix is summarized in the following proposition.

**Proposition II.1:** Consider the score function \( s(x) = \min_k W_{M_0}(f_\theta(x), y^k) \) as a function of \( f_\theta(x) \), it reaches the maximum when \( f_\theta(x) = (\frac{1}{K}, \ldots, \frac{1}{K}) \).

This proposition implies that the score function reaches the maximum when the classifier cannot determine which class the sample falls into. Also, combined with \( s(x) = 0 \) when \( f_\theta(x) = y^k \) (following axioms of a distance in (22), which is in Appendix A and available online), we can conclude that the loss function will train the classifier to map InD and OOD samples in opposite directions. For the InD samples, the model is trained to output softmax scores close to their labels, while for the OOD samples, the model is trained to output softmax scores that are away from any labels and close to \( (\frac{1}{K}, \ldots, \frac{1}{K}) \). This feature also lays the foundation for using the Wasserstein-based score function with the binary distance matrix as the loss function. The proof of Proposition II.1 is detailed in Appendix B, available online.

2) **Dynamic Distance Matrix:** Different from the binary distance matrix, we tailor a dynamic distance matrix for the OOD detection purpose. This distance matrix is calculated by the pair-wise absolute difference between elements in a predicted softmax score \( f_\theta(x) \) and a one-hot vector label \( y^k \), which is given as

\[
M_{Dy}(f_\theta(x), y^k) = \left[ f_\theta(x) - 0_K, \ldots, \frac{1}{K} - f_\theta(x), \ldots, f_\theta(x) - 0_K \right],
\]

(11)

where \( 0_K \in \mathbb{R}^K \) denotes the \( K \)-dimensional vector whose elements are all zeros. In (11), the dynamic distance matrix is denoted as a function of \( f_\theta(x) \) and \( y^k \), which will be dynamically changed with different marginal distributions. In the following parts, we use \( M_{Dy} \) for simplicity. The dynamic distance matrix is not a metric matrix, and hence, the Wasserstein distance does not satisfy the axioms of distance. However, the properties of the Wasserstein distance with a dynamic distance matrix make it a good fit for the OOD detection problem. First, the dynamic distance matrix \( M_{Dy} \) is invariant with respect to any one-hot vector \( y^k \), setting an equal penalty on the loss function when an OOD sample is close to any InD classes. The proposition is formally expressed as follows.

**Proposition II.2:**

\[
W_{M_0}(f_\theta(x), y^1) = \ldots = W_{M_0}(f_\theta(x), y^K)
\]

(12)

This proposition reveals the intuition of designing the dynamic distance matrix. Unlike the classification task, the OOD detection mainly cares about whether an incoming sample is OOD or not, which means the proposed method can regard all the InD classes as the same. Using the Wasserstein distance with a dynamic distance matrix, the minimal operator in the score function and loss function in (7) and (9) can be eliminated. Thus, the formulation is simplified as

\[
s(x) = \inf_P \langle P, M \rangle,
\]

(13)

\[
\mathcal{L}(\mathcal{D}_{InD}, \mathcal{D}_{OOD}) = \frac{1}{N_{InD}} \sum_{(x_{InD}, k) \in \mathcal{D}_{InD}} - \log \left( f_\theta(x_{InD})^\top y^k \right) - \beta \frac{1}{N_{OOD}} \sum_{x_{OOD} \in \mathcal{D}_{OOD}} \inf_P \langle P, M \rangle,
\]

(14)

where \( k \in K_{InD} \) is randomly selected because different class labels have no influence on the Wasserstein distance using the dynamic distance matrix.
The second proposition described below shows the maximum point of the Wasserstein-based score function with the dynamic distance matrix.

**Proposition II.3:** Consider \( W_{\theta}(f_\theta(x), y^k) \) as a function of \( f_\theta(x) \), it reaches the maximum when \( f_\theta(x) = \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) \).

Similar to Proposition II.1, this proposition implies that the Wasserstein-based score function defined upon the dynamic distance matrix also reaches the maximum when the classifier cannot determine which class the sample falls into. The combination of Propositions II.2 and II.3 captures the geometric shape of \( W_{\theta}(f_\theta(x), y^k) \) as a function of the softmax vector \( f_\theta(x) \): i.e. a function defined on a hyperplane, taking the maximal value at the center \( \left( \frac{1}{K}, \ldots, \frac{1}{K} \right) \), and taking the minimal value on the boundary \( y^k, k \in K_{\text{InD}} \). The proof of Propositions II.2 and II.3 are also included in Appendix B, available online.

### 3) Influence of Different Distance Matrices on Computational Complexity

The (13) and (14) indicate that compared with the binary distance matrix, using the dynamic distance matrix reduces the computational cost in calculating the WOOD loss and the score function because the minimal operator is eliminated. The score function \( s(x) \) is used as an example to theoretically illustrate the influence on computational complexity. Suppose there is a random input \( x \), if the dynamic distance matrix is used, the computational complexity of calculating the regularized Wasserstein distance between \( f_\theta(x) \) and \( y^k \) is \( O(K^2) \) [29] (introduced in Appendix A, available online). Thus, the computational complexity of \( s(x) \) with dynamic distance matrix \( (13) \) is \( O(K^2) \). In contrast, if the binary distance matrix is used, as shown in (7), the score function needs to find the minimum Wasserstein distance between the output softmax score and all \( K \) possible InD classes, which increases the computational complexity to \( O(K^3) \). A similar influence also exists when calculating WOOD loss with different formulations of the distance matrix. From the perspective of computational complexity, the dynamic distance matrix is preferably selected.

### III. COMPUTATIONAL AND THEORETICAL PERSPECTIVES

Given the formulation of WOOD, the following task is to learn the model parameters with WOOD loss. In this section, we will first show the procedure of model training and then discuss the statistical properties of the proposed method.

#### A. Gradient-Based Model Training

The gradient-based method is usually applied in deep learning to learn the model parameters. As shown in (9), the proposed WOOD loss consists of two components: the cross-entropy used for classifying InD samples and the Wasserstein-based score function for detecting OOD samples. The gradient of cross-entropy has been studied in the classification models [30]. Here, we focus on deriving the gradient of the Wasserstein-based score function. To do this, it is suffice to find a way to compute

\[
\nabla_\theta \min_k \inf_P \langle P, M \rangle.
\]

Under the following proposition (see proof in Appendix C, available online), the partial derivative and minimal operators can be interchanged.

**Proposition III.1:** Suppose \( \eta_i(\cdot) \)'s are mutually different at \( \theta \), and \( \eta_i(\theta) \) is first-order differentiable, with \( t^* = \arg \min_i \eta_i(\theta) \), we have \( \nabla_\theta \min_i \eta_i(\theta) = \nabla_\theta \eta_i(t^*) \).

That is to say:

\[
\nabla_\theta \min_k \inf_P \langle P, M \rangle = \min_k \nabla_\theta \inf_P \langle P, M \rangle.
\]

Note that \( P \) and \( M \) depend on \( k^* \) that minimizes the Wasserstein distance from the one-hot vector in class \( k \) to \( f_\theta(x) \). According to the chain rules,

\[
\nabla_\theta \inf_{P \in \Pi^*} \langle P, M \rangle = \nabla_{f_\theta(x)} \inf_{P \in \Pi^*} \langle P, M \rangle \times \nabla_\theta f_\theta(x),
\]

where \( \Pi^* \) is the set of joint distributions with \( f_\theta(x) \) and \( y^k \) as marginal distributions. Note that \( \nabla_\theta f_\theta(x) \) is the gradient of the softmax output of a deep neural network, which has been widely studied in the related literature. Therefore we only need to focus on the gradient term \( \nabla_{f_\theta(x)} \inf_{P \in \Pi^*} \langle P, M \rangle \), which is equivalent to the gradient of \( W(f_\theta(x), y^k) \) in (6).

We derive the gradient of Wasserstein distance with respect to \( f_\theta(x) \) following similar procedures in [25]. The basic idea is that Wasserstein distance (shown in (6)) is linear programming (LP), its gradient with respect to \( f_\theta(x) \) can be computed via Lagrangian duality. The optimal values of primal and dual problems are equal. So, the value of the corresponding dual variable at the optimal point is the desired gradient of the Wasserstein distance with respect to \( f_\theta(x) \). Suppose we have dual variables \( a, b \) corresponding to primal variables \( y^k, f_\theta(x) \), respectively, and two intermediate variables \( u, v \) are defined as \( u = \exp^{-\lambda a - \frac{1}{2}}, v = \exp^{-kb - \frac{1}{2}} \), the \( u \) and \( v \) can be solved iteratively by Sinkhorn-Knopp algorithm [29], which is given in Algorithm 2 (in Appendix D, available online). Given the value of \( v \) when converging as \( v^* \), the gradient of Wasserstein distance with respect to \( f_\theta(x) \) is given as

\[
\nabla_{f_\theta(x)} W(f_\theta(x), y^k) = b^* = -\frac{1}{\lambda} \left( \log v^* + \frac{1}{2} \right).
\]

The detailed derivation of the gradient is in Appendix D, available online. To this point, we can compute the gradient of WOOD loss with respect to the softmax scores of InD samples \( x_{\text{InD}} \) and OOD samples \( x_{\text{OOD}} \) as

\[
\nabla_{f_\theta(x_{\text{InD}})} \mathcal{L} = \frac{-1}{N_{\text{InD}}} \left( \frac{1}{f_\theta(x_{\text{InD}})} \right)^\top y^k, \quad (18)
\]

\[
\nabla_{f_\theta(x_{\text{OOD}})} \mathcal{L} = \frac{\beta}{\lambda N_{\text{OOD}}} \left( \log v^* + \frac{1}{2} \right), \quad (19)
\]

where \( v^* \) is calculated by Algorithm 2 with \( y^k \) and \( f_\theta(x_{\text{OOD}}) \) as the inputs. The gradient of WOOD loss with respect to the model parameters \( \theta \) can be further derived via the chain rule.

#### B. Training Algorithm

The training process using WOOD loss is summarized in Algorithm 1. \( X \) and \( K \) denote one batch of samples and their corresponding labels, respectively. In each batch, \( X_{\text{InD}} \) and \( Y_{\text{InD}} \)
## Algorithm 1: Training Algorithm of WOOD

1. **Inputs:** $\mathcal{D}_{\text{InD}}, \mathcal{D}_{\text{OOD}}$
2. **Initialize:**
   - $\beta$ \(\triangleright\) hyperparameter in loss function
   - $K$ \(\triangleright\) the number of class
   - $\text{ClsModel}$ \(\triangleright\) classification model
3. **Training in one epoch:**
4. **for** $x_{\text{Ind}} \in \mathbb{R}^{|B_{\text{Ind}}| \times C \times H_1 \times H_2}$, $y_{\text{Ind}} \in \mathbb{R}^{|B_{\text{Ind}}|}$ in $\mathcal{D}_{\text{Ind}}$ **do**
5. **Randomly select** $x_{\text{OOD}} \in \mathbb{R}^{|B_{\text{OOD}}| \times C \times H_1 \times H_2}$ in $\mathcal{D}_{\text{OOD}}$
6. **Forward Propagation:**
   - $y_{\text{OOD}} \in \mathbb{R}^{|B_{\text{OOD}}| \times K}$ \(\leftarrow\) indicate OOD samples
   - $x \leftarrow [x_{\text{Ind}}, x_{\text{OOD}}] \in \mathbb{R}((|B_{\text{Ind}}| + |B_{\text{OOD}}|) \times C \times H_1 \times H_2)$
   - $Y \leftarrow [Y_{\text{Ind}}, Y_{\text{OOD}}] \in \mathbb{R}((|B_{\text{Ind}}| + |B_{\text{OOD}}|) \times K)\)
   - $f_\theta(x) \leftarrow \text{softmax} (\text{ClsModel}(x))$
7. **The value of Loss:**
8. **for** $(x_{\text{Ind}}, f_\theta(x_{\text{Ind}})) \leftarrow (x_{\text{Ind}}, f_\theta(x_{\text{Ind}}))[\text{InD-idx}]$
9. **for** $(x_{\text{OOD}}, y_{\text{OOD}}) \leftarrow (x_{\text{OOD}}, f_\theta(x_{\text{OOD}}))[\text{OOD-idx}]$
10. **InD-Loss** $\leftarrow \frac{1}{|B_{\text{Ind}}|} \sum_{i} - \log (f_\theta(x_{\text{Ind}})^i y_i^k)$
11. **OOD-Loss** $\leftarrow \frac{1}{|B_{\text{OOD}}|} \sum_{i} \inf_{f_i} \text{log} (P_i, M_i)$
12. **L** $\leftarrow \text{InD-Loss} - \beta \times \text{OOD-Loss}$
13. **Backward Propagation:**
   - $\frac{\partial L}{\partial f_\theta(x_{\text{Ind}})} \leftarrow \frac{1}{|B_{\text{Ind}}|} \sum_{i} \frac{y_i^k}{f_\theta(x_{\text{Ind}})^i}$
   - $\frac{\partial L}{\partial f_\theta(x_{\text{OOD}})} \leftarrow \frac{\beta}{|B_{\text{OOD}}|} \frac{\sum_{i} \text{log} (P_i, M_i)^i}{|B_{\text{OOD}}|}$
14. **Update model parameters with gradients**


### Denote the portion of InD images and their labels, respectively. $x_{\text{OOD}}$ denotes the portion of OOD images in each batch. Labels of OOD images $y_{\text{OOD}}$ are only used to identify the OOD samples during training. Each image is denoted as a tensor with the dimension of $C \times H_1 \times H_2$, in which $C$ represents the number of channels, $H_1$ and $H_2$ represent the length and width, respectively. The number of InD and OOD samples in each batch is $|B_{\text{Ind}}|$ and $|B_{\text{OOD}}|$, respectively. The algorithm mainly consists of four steps: (i) in each iteration, a batch of InD samples is mixed with fewer OOD samples, and the label $1_K$ assigned to OOD samples is only used to identify them in the training phase and not used in calculation; (ii) in the forward propagation, the InD and OOD samples are fed into the classifier to generate the softmax scores, and the value of the WOOD loss is calculated by (9); (iii) in the backward propagation, the gradients of the WOOD loss with respect to the softmax scores are calculated by (18), (19), and the gradients with respect to model parameters are generated via chain rule; (iv) the model parameters are then updated via the gradient-based method.

### C. Statistical Properties of the WOOD Loss

In this section, we investigate the statistical properties of the WOOD loss. Suppose $(x_{\text{Ind}}, k) \in \mathcal{D}_{\text{Ind}}$ and $x_{\text{OOD}} \in \mathcal{D}_{\text{OOD}}$ are independent samples, $N_{\text{Ind}}$ and $N_{\text{OOD}}$ are the number of samples in $\mathcal{D}_{\text{Ind}}$ and $\mathcal{D}_{\text{OOD}}$, respectively, $f_\theta$ is the empirical risk minimizer, and $\mathcal{F}$ is the hypothesis space of functions mapping into $\mathbb{R}^K$, we have the statistical learning bound of the WOOD loss given in Theorem III.2.

**Theorem III.2:** For any $\delta > 0$, with probability at least $1 - \delta$, it holds that
\[
\mathbb{E}[\mathcal{L}(\mathcal{D}_{\text{Ind}}, \mathcal{D}_{\text{OOD}})] - \inf_{f_\theta \in \mathcal{F}} \mathbb{E}[\mathcal{L}(\mathcal{D}_{\text{Ind}}, \mathcal{D}_{\text{OOD}})] \\
\leq \frac{4\sqrt{\beta}}{m} \mathcal{R}_{N_{\text{Ind}}} (\mathcal{F}) + 2 \sqrt{(1 - m)^2 \text{log}(1/\delta)} + 2 \sqrt{\frac{\alpha^2 M \text{log}(1/\delta)}{2N_{\text{OOD}}}},
\]
where $\alpha_M$ is the maximum entry in all matrices $M$, $m$ is the minimum value in all predicted softmax scores $f_\theta(x_{\text{Ind}})$, $\mathcal{R}_{N_{\text{Ind}}}$ and $\mathcal{R}_{N_{\text{OOD}}}$ are Rademacher complexity [31].

The detailed derivation is provided in Appendix E, available online. The upper bound in Theorem III.2 guarantees that with a sufficient number of training samples, the difference between the minimized loss value and the global optimum is bounded by an arbitrarily small value, which indicates the loss value achieved by the empirical minimizer will approach the global optimum.

### IV. EXPERIMENTS

In this section, we introduce the experiment setup and demonstrate the performance of WOOD on different InD and OOD datasets. The implementation of our work is available at https://github.com/wyn430/WOOD.

#### A. Experiment Setup

1) **Datasets:** The datasets used in the experiments are MNIST [32], FashionMNIST [33], CIFAR-10 [34], CIFAR-100 [34], SVHN [35], downsampled TinyImageNet, and image-like data following Gaussian and Uniform distributions. TinyImageNet is a subset of ImageNet [36] which contains images with the shape of $3 \times 64 \times 64$ in 200 classes. To keep the image dimension of the TinyImageNet consistent with CIFAR-10 and SVHN, we further downsample it into the dimension of $3 \times 32 \times 32$ by resizing (TinyImageNet-r) or cropping (TinyImageNet-c). The basic information of these datasets, including the number of classes, image dimension, the number of training samples, and the number of testing samples are summarized in Table 1. These datasets are used as InD and OOD samples to validate the performance of WOOD, which is indicated in the first column of Table II.

2) **Hyperparameters:** Considering the proposed WOOD is a general framework to enable classifiers to detect OOD samples, we test its performance on the state-of-the-art classifier DenseNet [37] in the experiments. For the DenseNet, we follow the setup introduced in [37], with model depth 100, growth rate 12, and dropout rate 0.

$\beta$ is the hyperparameter used in WOOD loss to balance the focus of the classifier in classifying the InD samples and detecting the OOD samples. The optimal $\beta = 0.1$ is obtained in the experiments that enable the classifier to receive a good

---

The above text is the natural representation of the document, formatted in a readable and consistent manner. The table includes the algorithm and theorem for WOOD, along with explanations and derivations for the statistical properties of the WOOD loss. The experiments section outlines the datasets used and the hyperparameters tested.
TABLE I

| Datasets | Number of Class | Image Dimension | Number of Training | Number of Testing |
|----------|----------------|-----------------|-------------------|------------------|
| MNIST    | 10             | $28 \times 28$  | 60000             | 10000            |
| FashionMNIST | 10            | $28 \times 28$  | 60000             | 10000            |
| CIFAR-10 | 10             | $3 \times 32 \times 32$ | 50000             | 10000            |
| CIFAR-100| 100            | $3 \times 32 \times 32$ | 50000             | 10000            |
| SVHN     | 10             | $3 \times 32 \times 32$ | 73257             | 26302            |
| TinyImageNet-r | 200      | $3 \times 32 \times 32$ | 100000            | 10000            |
| TinyImageNet-c | 200   | $3 \times 32 \times 32$ | 100000            | 10000            |
| Gaussian | N/A            | $3 \times 32 \times 32$ | 10000             | 10000            |
| Uniform  | N/A            | $3 \times 32 \times 32$ | 10000             | 10000            |

TABLE II

| InD  | OOD    | FNR (95% TNR) ↓ |
|------|--------|-----------------|
|      |        | ODIN [8]        | Maha [10]        | ODIN* [9]       | Energy [20] | 1D [15] | WOOD-binary | WOOD-dynamic |
|      |        | ODIN [8]        | Maha [10]        | ODIN* [9]       | Energy [20] | 1D [15] | WOOD-binary | WOOD-dynamic |
| MNIST| FashionMNIST| 0.0589          | < 0.00001        | 0.0008          | 0.0005      | 0.0157  | < 0.00001   | < 0.00001   |
| FashionMNIST| 0.0287 | 0.1657          | 0.1045           | 0.0010          | 0.2428      | < 0.00001 | < 0.00001   | < 0.00001   |
| CIFAR-10 | SVHN    | 0.1693          | 0.1927           | 0.1017          | 0.0030      | 0.1765  | 0.0005      | 0.0011      |
|        | TinyImageNet-r | 0.0430 | 0.3635          | 0.2579          | 0.0505      | 0.6269  | 0.0095      | 0.0039      |
|        | TinyImageNet-c | 0.0001 | 0.3709          | 0.2568          | 0.0275      | 0.2128  | < 0.00001   | 0.0002      |
|        | Gaussian  | < 0.00001       | < 0.00001        | < 0.00001       | 0.0005      | 0.0088  | < 0.00001   | < 0.00001   |
|        | Uniform   | < 0.00001       | < 0.00001        | < 0.00001       | < 0.00001   | 0.0047  | < 0.00001   | < 0.00001   |
| CIFAR-100 | SVHN    | 0.4973          | 0.4082           | 0.1723          | 0.0235      | 0.5683  | 0.0202      | 0.0020      |
|        | TinyImageNet-r | 0.2524 | 0.3158          | 0.1818          | 0.5740      | 0.6962  | 0.0036      | 0.0058      |
|        | TinyImageNet-c | 0.2534 | 0.3209          | 0.1846          | 0.0975      | 0.6702  | 0.0042      | 0.0002      |
|        | Gaussian  | < 0.00001       | < 0.00001        | < 0.00001       | 0.0010      | 0.3299  | < 0.00001   | < 0.00001   |
|        | Uniform   | < 0.00001       | < 0.00001        | < 0.00001       | 0.0010      | 0.4028  | < 0.00001   | < 0.00001   |

InD  | OOD    | AUROC ↑ |
|------|--------|---------|
|      |        | ODIN [8] | Maha [10] | ODIN* [9] | Energy [20] | 1D [15] | WOOD-binary | WOOD-dynamic |
| MNIST| FashionMNIST| 0.9837  | > 0.99999 | 0.9990   | 0.9995    | 0.9995   | > 0.99999  | > 0.99999  |
| FashionMNIST| 0.9896   | 0.9666  | 0.9806   | 0.9995   | 0.9348    | > 0.99999 | > 0.99999  |
| CIFAR-10 | SVHN    | 0.9629  | 0.9515  | 0.9799  | 0.9983   | 0.9614  | 0.9987   | 0.9982     |
|        | TinyImageNet-r | 0.9910 | 0.9736  | 0.9529  | 0.9040  | 0.8851  | 0.9979  | 0.9978     |
|        | TinyImageNet-c | 0.9888 | 0.9249  | 0.9543  | 0.9918  | 0.9469  | > 0.99999 | 0.9996     |
|        | Gaussian  | 0.9999  | 0.9998  | 0.9996  | 0.9998  | 0.9981  | > 0.99999 | > 0.99999  |
|        | Uniform   | 0.9998  | 0.9999  | 0.9999  | 0.9997  | 0.9989  | > 0.99999 | > 0.99999  |
| CIFAR-100 | SVHN    | 0.8874  | 0.9158  | 0.9877  | 0.9944  | 0.8264  | 0.9950  | 0.9966     |
|        | TinyImageNet-r | 0.7993 | 0.8231  | 0.9667  | 0.8916  | 0.7464  | 0.9956  | 0.9959     |
|        | TinyImageNet-c | 0.8008 | 0.8163  | 0.9661  | 0.9783  | 0.7756  | 0.9950  | 0.9986     |
|        | Gaussian  | 0.9988  | 0.9999  | > 0.99999 | 0.9997  | 0.7987  | > 0.99999 | > 0.99999  |
|        | Uniform   | 0.9984  | 0.9999  | > 0.99999 | 0.9998  | 0.7355  | > 0.99999 | > 0.99999  |

performance in OOD detection without decaying its performance in classification. The value of $\beta$ is determined by the grid search with the range of [0,1] and the step size as 0.1. The impact of $\beta$ on the InD classification and OOD detection is discussed in Section IV-D. $B_{\text{InD}}$ and $B_{\text{OOD}}$ in Algorithm 1 are the batch sizes of InD and OOD samples in each training iteration. We set $B_{\text{InD}} = 50$ and $B_{\text{OOD}} = 20$ in training. $\epsilon$ in (8) is determined by the 95% TNR on InD testing samples.

3) Baseline Methods and Evaluation Metrics: To demonstrate the performance of the WOOD, five state-of-the-art OOD detection methods including ODIN [8], Maha [10], Generalized ODIN (ODIN*) [9], Energy-based Method (Energy) [20], and One-dimensional Method (1D) [15] are selected as the baseline methods.

All the OOD detection methods are compared by two evaluation metrics: (1) The FNR of OOD samples at 95% TNR, which indicates how many OOD samples are misidentified when the threshold $\epsilon$ is set to ensure 95% of InD samples are correctly identified. The FNR at 95% TNR is the lower, the better. (2) The area under the receiver operating characteristic curve (AUROC), which is the higher the better. In addition, all the methods except for 1D are designed to require exposure to the OOD samples. To keep a fair comparison, we keep the same pool of OOD samples accessible during the training or finetuning.
B. Results

The performances of the proposed WOOD and baseline methods are compared on 12 different combinations of InD and OOD datasets. The experiment results are summarized in Table II. In general, the WOOD method receives comparable performance when using binary and dynamic distance matrices, and both of them outperform the baseline methods consistently. More specifically, all the baseline methods show strength in detecting OOD samples from some datasets while failing in others. For example, Gaussian or Uniform distributed OOD data is the easiest scenario, and most of the baseline methods receive good performance except for applying the 1D method to detect them from CIFAR-100. ODIN generally works well in OOD detection on the small-scale classification problem (i.e., 10-class classification) while its performance apparently decayed when trying to identify SVHN, TinyImageNet-c, and TinyImageNet-r from the large-scale classification task (i.e., 100-class classification on CIFAR-100). ODIN* improved the performance on the OOD detection in the large-scale classification task, but has issues in identifying TinyImageNet-r and TinyImageNet-c from CIFAR-10. The energy method performs well in most of the scenarios with the struggle in identifying TinyImageNet-r from CIFAR-10 and CIFAR-100. It is also worth noting that the 1D method also receives good performance in some scenarios although it has no exposure to OOD samples during training. Compared with all the baseline methods, the proposed WOOD method improves both the FNR at 95% TNR and AUROC in OOD detection for both small-scale and large-scale classification tasks and receives consistent outstanding performance in identifying various types of OOD samples.

The score function in the WOOD is used to distinguish InD and OOD samples. Ideally, after training the classifier with WOOD loss, the values of the score function for the InD samples should be concentrated close to 0, while the values of the score function for the OOD samples should be concentrated away from 0. We demonstrate the histogram of the score function for different InD and OOD datasets in Fig. 3. The title of each subplot represents the InD and OOD datasets. For example “FashionMNIST/MNIST” represents the FashionMNIST is the InD dataset and MNIST is the OOD dataset. In Fig. 3, blue bars represent the histogram of score function for InD samples and red bars represent the histogram of score function for OOD samples. We can clearly figure out that after training the classifier with WOOD loss, the score function can successfully distinguish the InD and OOD datasets. This also explains why the WOOD framework consistently receives great performance in identifying all the OOD datasets.

As we discussed in Sections II-D1 and II-D2, when using the Wasserstein-based score function in WOOD loss, for the OOD samples, the classifier is trained to output predicted probability away from any labels and close to \( \frac{1}{K}, \ldots , \frac{1}{K} \). To validate this property, we randomly select one OOD sample and one InD sample and generate the predicted probabilities from the trained classifier, which is shown in Fig. 4. We can find out that when using WOOD loss with dynamic distance matrix, the predicted probability of the InD sample is close to its label (the score function close to 0) and the predicted probability of the OOD sample is close to \( \frac{1}{10} \ldots \frac{1}{10} \) (the score function close to the maximum). This property also indicates that the value of the WOOD loss for the trained classifier converges to the minimum value, in which the first term (cross-entropy loss) approaches 0 and the second term (Wasserstein-based score function) approaches the maximum value determined by the distance matrix. When using WOOD loss with the binary distance matrix, the trained classifier has a similar property. This property also reveals the flexibility to select the score function for detection after training with the WOOD loss. As long as the selected score function can distinguish the predicted probability close to \( \frac{1}{K}, \ldots , \frac{1}{K} \) versus the probability close to any InD label in its defined space, it can be used in a threshold-based detector after training with WOOD loss. We included the example to support this conclusion in Appendix F, available online.

C. Selection of Distance Matrix in WOOD

From the results shown in Table II, we can conclude that there is no significant difference in the quantitative performance of WOOD with different distance matrices. As we discussed in Section II-D3, theoretically, the dynamic distance matrix will reduce the computational complexity of the Wasserstein-based score function from \( O(K^3) \) to \( O(K^2) \). We also validate this property in the experiment. To eliminate the influence from
other computations, such as the calculation of cross-entropy loss, back-propagation, etc., we use the calculation of the Wasserstein-based score function for the single image in CIFAR-10 as an example. When using the dynamic distance matrix, the average computational time of the Wasserstein-based score function for 100 samples is 0.0035 seconds. In comparison, when using the binary distance matrix, the average computational time of the Wasserstein-based score function for 100 samples is 0.0307 seconds. Given $K = 10$ in CIFAR-10, the improvement of computational complexity is consistent with the theoretical analysis. Moreover, such improvement will be more significant when the number of InD classes $K$ further increases.

We also validated it on OOD detection for classifying CIFAR-100. The average computational time of the Wasserstein-based score function for 100 samples is 0.0038 seconds when using the dynamic distance matrix and 0.3361 seconds when using the binary distance matrix. Increasing the scale of the classification task does not impact the computational cost when using the dynamic distance matrix. Thus, if the Wasserstein distance is specifically used in OOD detection, we recommend using the binary distance matrix when $K$ is large.

**D. Hyperparameter Selection**

$\beta$ is the hyperparameter introduced in the proposed WOOD loss to balance the InD classification and OOD detection. The value of $\beta$ ranges within $[0,1]$. Intuitively, a small value of $\beta$ indicates the training process will focus more on the InD classification while a large $\beta$ will focus more on the OOD detection. To validate this intuition and also provide insights on selecting the value of $\beta$, we demonstrated the change in InD classification accuracy (bottom subplot in Fig. 5) and OOD detection AUROC (top subplot in Fig. 5) with different values of $\beta$.

In Fig. 5, WOOD loss with dynamic distance matrix is applied to distinguish SVHN, TinyImageNet-r, and TinyImageNet-c from CIFAR-10 using 11 different values of $\beta$. The reference accuracy denoted in the black dashed line indicates the InD classification accuracy only with the first term (cross-entropy loss) in WOOD loss. From Fig. 5, we can conclude that a large value of $\beta$ is more likely to incur a decay in InD classification accuracy, which is apparent when having TinyImageNet-r as the OOD samples. Thus, it is recommended to tune the value of $\beta$ from small values (i.e., $\beta = 0.1$) when applying the proposed framework to new scenarios.

It is worth noting that the Wasserstein distance with the binary distance matrix is not limited to be used in the OOD detection task and can be more generally used as the loss function in other computer vision tasks, such as object classification and detection [25]. Thus, if the Wasserstein distance is used in OOD detection and other tasks simultaneously, we would recommend using the binary distance matrix to keep the distance measure consistent. This paper focuses on OOD detection and tests both matrices in the experiments. In our experiments on InD datasets with 10 classes the difference in computational complexity is not significant when using these two matrices.
V. CONCLUSION

OOD detection is a crucial task in protecting DNNs from cyber attacks. It can also improve the system’s resilience and security. This paper proposes a Wasserstein-based out-of-distribution detection (WOOD) method to strengthen the classifiers with the ability to identify OOD samples. The performance of the proposed method is validated by different combinations of InD and OOD datasets and demonstrated outstanding results compared with other OOD detection methods. The advantages of the WOOD method can be summarized into five aspects: (1) It is generally compatible with various classifiers and has little influence on the model complexity and training time; (2) The designed WOOD loss function for training the classifiers well follows the human’s intuitive rationales and intelligence in identifying OOD samples, i.e., trying to keep OOD samples away from InD samples instead of trying to assign a specific label to them; (3) The Wasserstein distance fully exploits the dissimilarity between output discrete distributions for InD and OOD samples; (4) The specifically designed dynamic distance matrix reduces the computational complexity of WOOD loss and the score function; (5) The analysis of statistical learning bound provides the theoretical guarantees in training classifiers with the proposed WOOD loss function.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in Advances in Neural Information Processing Systems, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger Eds., vol. 25, Red Hook, NY, USA: Curran Associates, Inc., 2012.

[2] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2016, pp. 770–778.

[3] K. He, G. Gkioxari, P. Dollár, and R. Girshick, “Mask R-CNN,” in Proc. IEEE Int. Conf. Comput. Vis., 2017, pp. 2980–2988.

[4] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[5] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 22, no. 10, pp. 1533–1545, Oct. 2014.

[6] L. Rüschendorf, “The Wasserstein distance and approximation theorems,” Probability Theory Related Fields, vol. 70, no. 1, pp. 117–129, 1985.

[7] D. Hendrycks and K. Gimpel, “A baseline for detecting misclassified and out-of-distribution examples in neural networks,” in Proc. 5th Int. Conf. Learn. Representations, Toulon, France, Apr. 24–26, 2017.

[8] S. Liang, Y. Li, and R. Srikant, “Enhancing the reliability of out-of-distribution image detection in neural networks,” in Proc. Int. Conf. Learn. Representations, 2018.

[9] Y.-C. Hsu, Y. Shen, H. Jin, and Z. Kira, “Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data,” in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit., 2020, pp. 951–960.

[10] K. Lee, K. Lee, H. Lee, and J. Shin, “A simple unified framework for detecting out-of-distribution samples and adversarial attacks,” in Advances in Neural Information Processing Systems, vol. 31, Red Hook, NY, USA: Curran Associates, Inc., 2018.

[11] T. Denouden, R. Salay, K. Czarnecki, V. Abdelzad, B. Phan, and S. Vernekar, Improving reconstruction autoencoder out-of-distribution detection with mahalanobis distance, 2018, arXiv:1812.02765.

[12] Y. Bahat and G. Shakhnarovich, “Confidence from invariance to image transformations,” 2018, arXiv:1804.00657.
Yinun Wang (Member, IEEE) received the BS degree in electrical engineering and automation from Xi’an Jiaotong University, Xi’an, China, in 2017, the MS degree in electrical engineering from Columbia University, New York, NY, USA, in 2019, and the PhD degree in industrial and systems engineering from Virginia Tech, Blacksburg, VA, USA, in 2022. Currently, he is an assistant professor with the Department of Industrial and Systems Engineering, Rensselaer Polytechnic Institute (RPI), Troy, USA. His research interests include data analytics, system pattern recognition, and machine learning techniques in quality control of advanced manufacturing systems. He is a recipient of FTC Early Career Award, 7 best paper awards (e.g., IEEE Transactions on Automatic Control, Best Paper Award), and two best dissertation awards. He also won the Faculty Achievement Award from RPI he is a member of the Institute of Industrial and Systems Engineers (IISE), the Institute for Operations Research and the Management Sciences (INFORMS), and the American Statistical Association (ASA).

Jionghua (Judy) Jin received the PhD degree from the University of Michigan, MI, USA, in 1999. She is currently a professor with the Department of Industrial and Operations Engineering, the University of Michigan. Her major research interests are data analytics and quality engineering for improving complex system operations and quality. Her research emphasizes a synergistic fusion of engineering models and simulations to the development of advanced data fusion and analytics methods across statistics, machine learning, system control, and optimization methods. Her research has received numerous awards including the Presidential Award (NSF-PECASE), NSF-CAREER Award, and fourteen Best Paper Awards. She is an elected fellow of ASME, IEEE, and INFORMS.

Xiaowei Yue (Senior Member, IEEE) received the BS degree in mechanical engineering from the Beijing Institute of Technology, Beijing, China, in 2004. He is currently a professor with the Grado Department of Industrial and Systems Engineering, University of Wisconsin–Madison, Madison, WI, USA, in 2004. He is currently a professor with the Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, VA, USA. His research interests include sensing and analytics for smart manufacturing and modeling, synthesis, and diagnosis for large and complex manufacturing systems. He is a fellow of the Institute of Industrial and Systems Engineers (IISE).