Adversarial Example Detection Using Latent Neighborhood Graph

Ahmed Abusnaina[†◇], Yuhang Wu[‡], Sunpreet Arora[‡], Yizhen Wang[‡],
Fei Wang[‡], Hao Yang[‡], and David Mohaisen[†¶]
† University of Central Florida, ‡ Visa Research
◇ ahmed.abusnaina@knights.ucf.edu, ¶ mohaisen@ucf.edu,
‡{yuhawu, sunarora, yizhewan, feiwang, haoyang}@visa.com

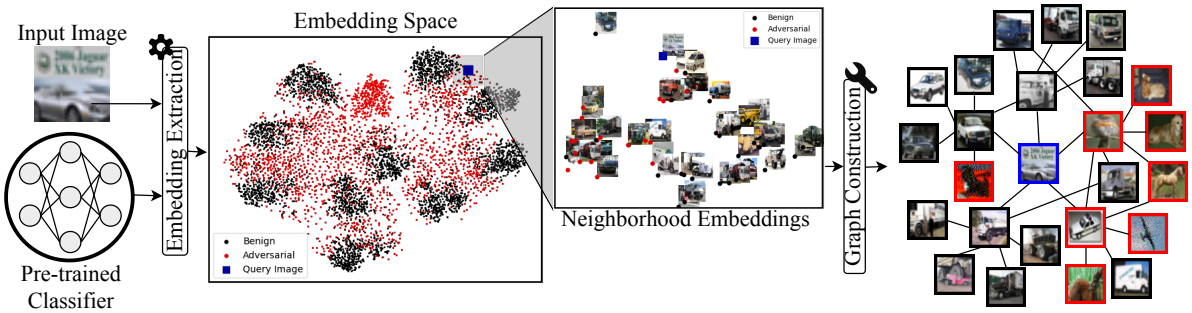


Figure 1: Generation of Latent Neighborhood Graph (LNG) for adversarial example detection. After computing the input example embedding, a LNG that describes the local manifold around the input example is constructed using both adversarial and benign example embeddings from a reference database. The LNG is then classified using a graph discriminator to determine whether the graph is generated from an adversarial or benign example.

Abstract

Detection of adversarial examples with high accuracy is critical for the security of deployed deep neural network-based models. We present the first graph-based adversarial detection method that constructs a Latent Neighborhood Graph (LNG) around an input example to determine if the input example is adversarial. Given an input example, selected reference adversarial and benign examples (represented as LNG nodes in Figure 1) are used to capture the local manifold in the vicinity of the input example. The LNG node connectivity parameters are optimized jointly with the parameters of a graph attention network in an end-to-end manner to determine the optimal graph topology for adversarial example detection. The graph attention network is used to determine if the LNG is derived from an adversarial or benign input example. Experimental evaluations on CIFAR-10, STL-10, and ImageNet datasets, using six adversarial attack methods, demonstrate that the proposed method outperforms state-of-the-art adversarial detection methods in white-box and gray-box settings. The proposed method is able to successfully detect adversarial examples crafted with small perturbations using unseen attacks.

1. Introduction

Deep learning techniques are being widely used in various domains including computer vision [6, 13, 10, 24], natural language processing [12, 43], and speech recognition [16, 37]. However, an extensive line of research has shown that an attacker can manipulate the prediction of a deep learning-based classification system by adding a small perturbation to deep learning model inputs, intermediate embeddings [19, 11, 8], or by inducing distribution shifts [26, 41]. These results highlight a major security issue for deep neural network-based prediction systems, especially the ones deployed in critical applications such as access control and user authentication [49].

To address this security concern, a variety of defense mechanisms have been proposed. These defense mechanisms can be broadly categorized into two categories. The *proactive* approaches, e.g. adversarial training [7, 14] and robustness-driven regularization [42], explicitly considers the presence of known adversarial attack methods to train a model, which increases model robustness to adversarial perturbation. However, in order to use this approach, existing models need to be re-trained, which can be costly. In contrast, the *reactive* approach requires no re-training of

existing models; instead, it builds a detector to filter adversarial examples in the test environment, and thus becomes a viable solution for already deployed systems. In addition, detection-based defense mechanisms can also help to identify security-compromised input sources.

A key finding of recent state-of-the-art detection methods [47, 38] is that there is a significant correlation between the legitimacy of an input example and its neighborhood information in the learned embedding space. For instance, the Deep k-Nearest Neighbors (DkNN) [47] detector computes the embedding of nearest neighbors of the input example at each layer of the network, and subsequently uses both the embedding and the class labels of the nearest neighbors to determine if the input is adversarial. Inspired by this insight, we propose a method to leverage dynamically constructed neighborhood graphs for detecting adversarial examples. We introduce *Latent Neighborhood Graph* – a general structure encoding not only the neighbors of the input, but also the *relation* between them – to represent the neighborhood of the input. Compared to DkNN, the benefits of our solution are three folds: (i) LNG covers multi-hop neighbors which characterizes the local manifolds of the input example, while DkNN only describes the manifold of the input example, (ii) LNG aggregates neighborhood information adaptively based on the connectivity learned on the embedding space which encodes much richer information than the class labels employed in DkNN, (iii) LNG incorporates both adversarial and benign neighbors in detection while DkNN only utilizes benign neighbors due to the measurement of consistency of the neighborhood labels at each layer of the network. In addition to information encoding, existing detectors are also limited by the computation cost. PeerNet [21], a graph-based convolutional network claimed to be robust to adversarial attacks, relies on pixel-wise neighborhood retrieval based on the intermediate 2D feature maps of a deep neural network, which increases the computation burden at test time. To overcome the aforementioned limitations, our approach purely relies on the embeddings at the final hidden layer of a deep neural network. We show that a combination of graph attention net and our novel LNG representation suffice to achieve state-of-the-art adversarial example detection performance.

In the proposed method, input example is used as a central node to construct a latent graph connected with samples curated from a reference dataset (see Figure 1). The graph describes the local manifold patterns for both the input example and its immediate benign and adversarial neighbors for adversarial detection. Both the nodes and the linkage of the graph are estimated on-the-fly, and we train the graph constructor and discriminator in an end-to-end manner. Experimental evaluations on three benchmark datasets show that the proposed approach yields state-of-the-art adversarial example detection performance against various known and unknown adversarial attacks, while maintaining high performance (more than 80%) against best-efforts white-box attack configuration.

The contributions of this work are as follows:

- We present the first work that poses adversarial example detection as a graph classification problem. Our method efficiently constructs a latent neighborhood graph using reference examples for adversarial example detection.
- The proposed method estimates the latent neighborhood graph's adjacency matrix on-the-fly based on the distances of neighborhood examples, and adaptively aggregates the information from both benign and adversarial neighbors for adversarial example detection.
- State-of-the-art gray-box and white-box detection performance on adversarial examples generated using known and unseen adversarial example generation methods.

2. Related Work

A variety of proactive defense techniques have been proposed to counter adversarial examples. Some of the earliest ones include adversarial training [19, 7, 14, 35, 28], gradient masking [14], distillation networks [29], feature squeezing [36], and k-NN search [4, 48]. Reactive approaches, on the contrary, aim to effectively learn to distinguish between benign and adversarial examples [20, 44, 46, 18, 9]. For example, Feiman *et al*. [32] develop a logistic regression-based (LR) adversarial example detector that uses kernel density and Bayesian uncertainty features. Ma *et al*. [38] estimate a Local Intrinsic Dimensionality (LID) score at each neural network layer using extreme value theory, and characterize key properties of the adversarial subspace for adversarial example detection. Ma *et al*. [34] analyzed the deep neural networks internals (*i.e*. weights) and proposed a network invariants, including value invariants and provenance invariants, extraction technique for adversarial example detection. Even though the aforementioned methods achieve competitive gray-box adversarial example detection accuracy, most of them can be circumvented using Carlini and Wagner (CW)'s optimization-based attack [1]. Recently, Hu *et al*. [33] proposed an algorithm that demonstrated empirical robustness to the CW attack. The two key steps used in this algorithm are: (i) the application of Gaussian noise on the input example, and (ii) the use of the number of steps required to change the classification of the example (from benign to adversarial and vice versa) as a distance metric to counter the powerful CW attack.

Another family of approaches uses nearest neighbors for adversarial defense. Deep k-Nearest Neighbors (DkNN) [47] method uses a k-nearest neighbor model at every layer of the network to assess if the input example is adversarial. Nearest neighbors, especially those that do not belong

Table 1: Comparison of different adversarial detection methods based on information used for defense. Use of (i) adversarial examples (Adv. Ex.) for training, (ii) input embedding space (Embedding) for prediction, (iii) detection is independent from the sample class (Class-indep.), and (iv) Graph-based adversarial detection (Graph).

Method	Adv. Ex.	Embedding	Class-indep.	Graph
Adv. training [14]	✓	×	×	×
Cohen *et al*. [15]	✓	×	✓	×
Mahalanobis [23]	✓	✓	×	×
DkNN [47]	×	✓	×	×
Hu *et al*. [33]	✓	✓	×	×
LID [38]	✓	✓	✓	×
Ours	✓	✓	✓	✓

to the majority class, are used for this determination. Kimin *et al*. [23] proposed a Mahalanobis distance-based method that models the distribution of samples in each class independently. Compared to [47, 23], our class-independent method does not make any prior assumption on the data distribution of a specific class and are less sensitive to the number of samples in each class. Recently, Svoboda et al. [21] propose PeerNets, a deep network structure that aggregates information from nearest neighbors to improve the robustness to adversarial attacks, and Cohen et al. [15] used the influence functions to identify important examples from a training dataset for the adversarial example detection task, and an LR classifier for predicting if the input example is adversarial. While such approaches (e.g., [47, 21, 15]) yield competitive detection performance, they suffer from high computational complexity: [47] requires the retrieval of nearest neighbors from a subset of deep network layers, [21] retrieves neighbors for each pixel on multiple 2D feature maps, and the method in [15] computes influence functions for the entire training dataset online.

Table 1 compares key differences in adversarial detection methods based on the information used for detection.

3. Methodology

Our defense mechanism first generates a latent neighborhood graph (LNG) for each input example, and then uses Graph Neural Networks (GNNs) to exploit the relationship between nodes in the neighborhood graph to distinguish between benign and adversarial examples. The fundamental premise is to harness the rich information in local manifolds with LNG, and use the GNNs model – with its high expressiveness – to effectively find higher-order patterns for adversarial example detection from the local manifolds of the nodes encoded in the graph.

Figure 2 shows the overview of our defense mechanism. First, for each image I in the data set, we extract its embedding $\mathbf{z}$ from the pre-trained neural network model we are defending, and use the embedding representation there-

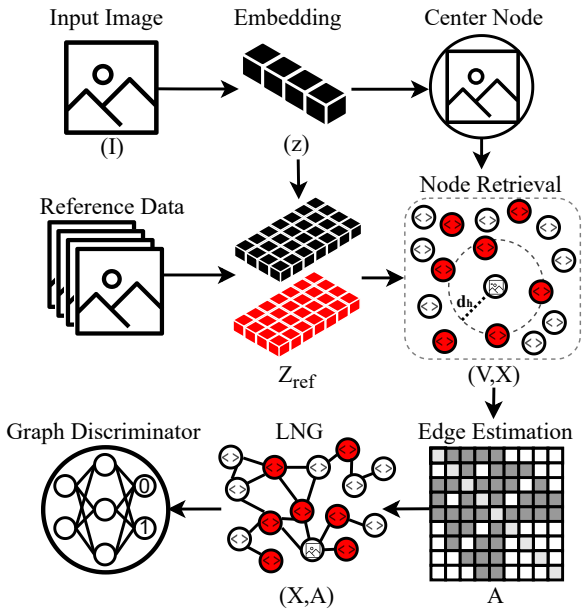


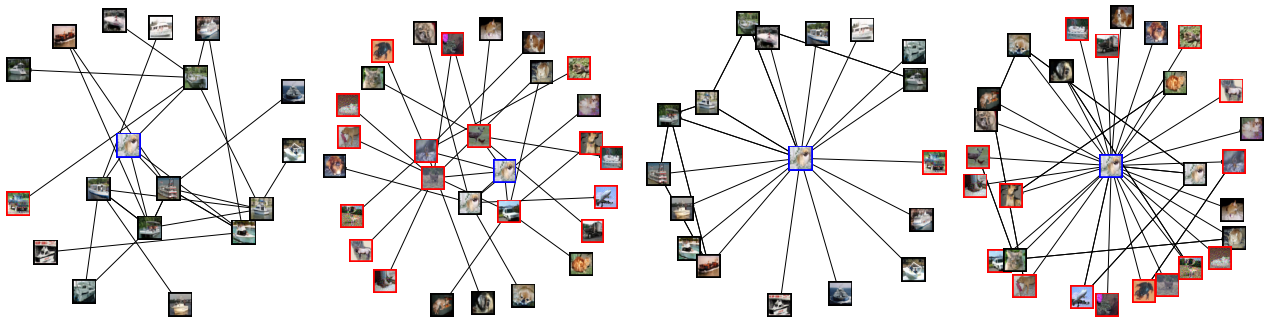
Figure 2: Overview of the proposed method.

after instead of the original pixel values. In addition to the training data for the original learning task, we maintain an additional *reference* data set for retrieving the manifold information. A neighborhood of n reference examples is selected around $\mathbf{z}$ from the reference set. After retrieving the reference examples, we construct the following two matrices: the $n \times m$ *embedding* matrix X stores the embeddings of neighborhood examples, where each row is a $1 \times m$ embedding vector of one example; the $n \times n$ *adjacency* matrix A encodes the manifold relation between all pairs of examples in the neighborhood. Since A is unknown, we propose an efficient algorithm to estimate A based on the embedding distance in the following sections. The LNG of $\mathbf{z}$ is characterized by these two matrices. Finally, a GNN model ingests both X and A as inputs, and predicts whether $\mathbf{z}$ is an adversarial example.

In the following, we explain the main components of our mechanism in detail. We first describe the creation of *reference* dataset, followed by generation of LNG, and the structure of our GNN model for adversarial example detection.

3.1. Reference Dataset

Given a training set of inputs Z, we randomly sample a subset of inputs Z_{ref}. We call such Z_{ref} the *clean* reference set because the inputs are all natural. Given a trained model for the original task, we can also create an *adversarially-augmented* reference set: we first pick an attack algorithm, create adversarial examples for all inputs in Z_{ref} against the given model, and add the adversarial examples to Z_{ref}. The resulted adversarially-augmented ref-



(a) Benign k-NNG (b) Adversarial k-NNG (c) Benign LNG (d) Adversarial LNG

Figure 3: Sample graphs generated by the proposed method. (a) and (c) show the k-NNG and LNG for the benign image, while (b) and (d) show the k-NNG and LNG for the adversarial pair generated using the same image. Blue border refers to the input image, while black and red borders refer to benign and adversarial neighbors, respectively.

erence set will have twice as many points as the clean reference set. We observe these adversarial samples are able to encode information regarding the layout of adversarial examples to benign examples in the local manifold.

3.2. Latent Neighborhood Graph

A latent neighborhood graph is characterized by an embedding matrix X and an adjacency matrix A. We construct an LNG by a 2-step procedure – node retrieval followed by edge estimation. The node retrieval process selects a set of points V in $\mathbf{z}$'s neighborhood from the reference data set. Stacking the embedding vectors of these points (including $\mathbf{z}$) yields the embedding matrix X. Edge estimation uses a data-driven approach to determine the relationships between nodes in V, which yields the adjacency matrix A.

3.2.1 Node Retrieval

The construction of V starts with the k-nearest-neighbor graph (k-NNG) of the input $\mathbf{z}$ and the nodes in Z_{ref}: each point in $Z_{ref} \bigcup \{\mathbf{z}\}$ is a node in the graph, and an edge from node i to node j exists *iff* j is among i's top-k nearest neighbors in Euclidean distance over the embedding space. We then keep the nodes whose *graph* distance from $\mathbf{z}$ in the k-NNG is within a threshold l. For example, if $l = 1$, then we only keep the immediate top-k nearest neighbors of $\mathbf{z}$ (one-hop neighbors); if $l = 2$, then we also keep the k nearest neighbors for each $\mathbf{z}$'s one-hop neighbors. Finally, we form V with n neighbors to $\mathbf{z}$. Based on this breadth-first-search strategy to construct V, the node retrieval method discovers all nodes with a fixed graph distance to $\mathbf{z}$, repeats the same procedure with increased graph distance until the maximum graph distance l is reached, and then returns the n neighbors to $\mathbf{z}$ from the discovered nodes.

Our approach can harness manifold information that is otherwise not possible using Euclidean distance, e.g. the Swiss-roll scenario [22]. We also note that when $n = k$, the node retrieval process is equivalent to selecting n nearest neighbors, like DkNN. Therefore, our approach offers more flexibility in learning the local manifold.

3.2.2 Edge Estimation

Next, we determine the edges of the LNG based on the nodes of k-NNG. The edges are paths to control the information aggregation across the graph, which creates the context to determine the center node's class. Since each node's embedding is extracted independently, it is important to let the system automatically determine the context used for adversarial detection, and also aware of the pair-wise relation between the query example and its neighbors. Motivated by the design of Cosmo *et al*. [25], we connected all the nodes in the generated graph with the center node using direct linking and adopt a data-driven approach to re-estimate the connections between neighbors. In particular, we model the relation between two nodes i, j as a sigmoid function of the Euclidean distance between them:

$$A_{i,j} = \frac{1}{1 + \exp(-t \cdot d(i,j) + \theta)},$$

where $d(i,j)$ is the Euclidean distance between i and j, and t, θ are two constant coefficients. Instead of manually assigning the coefficients t and θ, we make them learnable parameters and optimize them in an end-to-end manner with the graph discriminator introduced in the next section.

Figure 3 shows the k-NNG and LNG for a benign and its corresponding CW-based adversarial "dog" image from the CIFAR-10 dataset. The neighborhood nodes are highly related to the input image embedding, while the connections of LNG are estimated using the proposed approach.

3.3. Graph Discriminator

We use a specific graph attention network architecture [31] to aggregate information from $\mathbf{z}$ and its neighbors,

and at the same time learn the optimal t and θ to create the right context from $\mathbf{z}'s$ neighbors for adversarial detection. The network takes two inputs: the embedding matrix X and the adjacency matrix A of the latent neighborhood graph. The graph attention network architecture consists of four consecutive graph attention layers, followed by a dense layer with 512 neurons, and a dense classification layer with two-class output. Formally, let f denote a function in the model class, and let $X_{\mathbf{z}}$ and $A_{\mathbf{z}}$ denote the embedding and adjacency matrix of an input $\mathbf{z}$ generated by our LNG algorithm. During the training stage, we solve:

$$f^* = \arg\min_f \sum_{(\mathbf{z},y)} \ell(f(A_{\mathbf{z}}, X_{\mathbf{z}}), y)$$

where ℓ is the cross-entropy loss between the class probability prediction and the true label. To summarize, method can characterize the local manifold with LNG, and can adapt to different local manifolds based on graph attention network. Both factors are vital to our choice of using a GNN structure, and the empirical improvement of detection rates in Section 4 validates our belief.

4. Experiments

The proposed adversarial example detection approach is evaluated against six state-of-the-art adversarial example generation methods: FGSM (L_∞), PGD (L_∞), CW (L_∞), AutoAttack (L_∞), Square (L_∞), and boundary attack. All attacks are implemented as "non-targeted" attacks on three datasets: CIFAR-10 [5], ImageNet dataset [6], and STL-10 [3]. The non-targeted attacks are typically harder than the targeted attacks to detect, as less perturbation is applied. The performance is compared to four state-of-the-art adversarial examples detection approaches, namely DkNN [47], kNN [4], LID [38], as well as Hu *et al*. [33].

DkNN [47]: checks the label consistency of neighborhood examples in each deep network layers to test whether the input example is "off-manifold".

kNN [4]: shares the same intuition with DkNN. However, because it was originally proposed to work on a web-scale database, it uses fewer layers than DkNN. We converted this approach into an adversarial detector and employed the embedding layer for the nearest neighbors' retrieval.

LID [38]: characterizes properties of adversarial examples, which can be facilitated to detect adversarial examples when accompanied with a simple k-NN classifier.

Hu *et al*. [33]: is one of the most recent algorithms for adversarial detection, and was demonstrated to be extremely robust to white-box adversarial attacks. The method relies on an online search stage to measure the distance of the input example to a decision boundary.

4.1. Experimental Setup

Training and Testing. The CIFAR-10 dataset is split into three subsets, *training set* (45,000 images), *reference set* (5,000 images), and *testing set* (10,000 images). For ImageNet [30] dataset, we use the reference dataset of the 2012 original set, which contains a total of 50,000 labeled images (50 images per class). The dataset is split into two subsets, *reference set* (40,000) and *testing set* (10,000). For STL-10 dataset, we split the labeled images into three sets: *training set* (4,000 images), *reference set* (1,000 images), and *testing set* (8,000 images).

Table 2: Detection performance (AUC) of the k-NNG discriminator for different number of neighbors (k).

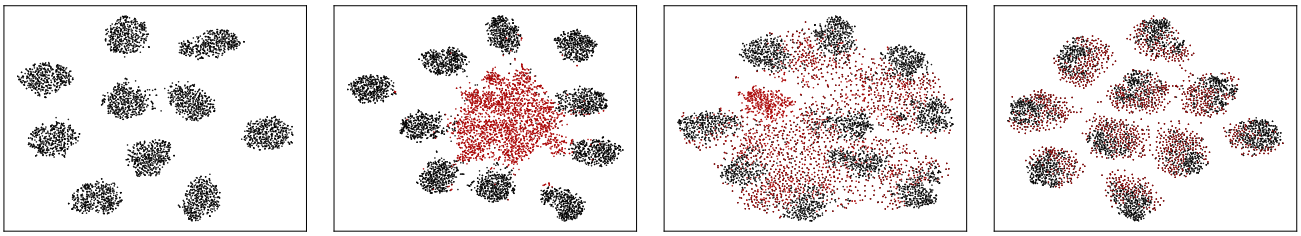
Neighbors				
2	3	4	**5**	6
96.39%	98.86%	99.23%	**99.54%**	99.17%

The ResNet-110 [40] classifier is trained on the CIFAR-10 *training set* and yields a classification accuracy of 93.41%. A pre-trained Densenet-121 [17] model with embedding of size 1×1024 and a reported accuracy of 75% is used for ImageNet. For STL-10 dataset, ResNet-20 classifier with classification accuracy of 82.30% is used. Any *reference* or *testing* examples incorrectly classified by the classifier were discarded.

The discriminator is trained on graphs generated using the *reference* dataset (see section 3) and adversarial examples generated using one adversarial attack method on the same dataset. We evaluate the performance of the discriminator using 100 random examples per class from the *testing* dataset for CIFAR-10, and the whole *testing* dataset for ImageNet and STL-10. Adversarial Robustness Toolbox [27] is used for implementing the adversarial attacks. For the baseline evaluation, we follow the same configurations used in the original DkNN [47] approach. For Hu *et al*.'s [33] and LID [38] adversarial detectors, the *reference set* is used to determine the thresholds that provide the best detection performance. All baseline adversarial detectors are trained on the reference dataset augmented with adversarial examples, and evaluated on the *test* set similar to our discriminator.

Parameter Tuning. To demonstrate the efficiency of the proposed method, we select $l = 2$ for k-nearest neighbor-based graph generation (section 3.2.1). To determine k, a line search is used. Table 2 shows the effect of changing k on the FGSM adversarial examples detection performance on the CIFAR-10 dataset using k-NNG. We set $k = 5$ in our approach considering the trade-off between benign and adversarial accuracy. To find the optimal number of neighbors in DkNN and kNN, we tested $k \in [10, \frac{(|reference\ set|)}{(\#classes)}]$ and then select $k = 200$ for CIFAR-10 dataset, and $k = 40$ for STL-10 and ImageNet datasets.

Feature space. To obtain the node feature for each neighborhood example, we use the image embedding generated



(a) Benign (b) FGSM (c) CW (d) CW_{wb}

Figure 4: A visualization of the embedding space (using t-SNE [45]) of (a) benign (black), and adversarial (red) examples generated using different adversarial attacks (b) FGSM, (c) Carlini and Wagner, and (d) white-box Carlini and Wagner on the CIFAR-10 dataset. Note specifically the considerable overlap of adversarial and benign clusters in (d).

using the pre-trained classifier. The DkNN classifier is evaluated using the outputs from all blocks (per block output for ResNet), while the kNN classifier is evaluated using the output of the embedding layer only. Originally, DkNN is used for adversarial examples recognition. However, by using the characteristics extracted, a simple LR detector can be trained for adversarial examples detection as shown in [15].

For LNG, the entries in A derived from the sigmoid function are real numbers in $[0, 1]$. We quantize the entries with a threshold value t_h as follows:

$$A'_{i,j} = \begin{cases} 0, & \text{if } A_{i,j} < t_h \\ 1, & \text{if } A_{i,j} \geq t_h \end{cases} .$$

The resulted binary A' is the final adjacency matrix of our LNG. Since the sigmoid function is monotonic w.r.t. $d(i, j)$, the threshold t_h will also correspond to a distance threshold d_h. A' implies that an edge exists between pairs of nodes closer than d_h. In practice, we line search t_h and choose the best value in validation.

4.2. Threat Model

The proposed method is evaluated in both the white-box and gray-box settings. In the following, a brief description of each setting is provided below.

White-box Setting. In this setting, the adversary is aware of the different steps involved in the adversarial defense method but does not have access to the method's parameters. Additionally, it is assumed that the entire *training* and *reference* sets are available to the adversary. To implement the white-box attack, the attack strategy of Carlini and Wagner [1] is used. The objective function of the CW minimization is modified as follows:

$$\underset{I_{adv}}{\text{argmin}} ||I_{adv} - I||_2^2 + c \cdot (l_{CW}(I_{adv}) + l_d(D(I_{adv}))),$$

where l_{CW} is the original adversarial loss term used in Carlini and Wagner [2], and $D(I_{adv})$ is negative of the summation of the distances between the adversarial example and each adversarial example in the constructed nearest neighbor graph, defined as:

$$l_d(D(I_{adv})) := -\sum D_{adv} = \sum_{i=1}^{N} \begin{matrix} 0;\ x(v_i) \in X_D \\ ||x_{adv} - x(v_i)||;\ x(v_i) \in X_{Dp} \end{matrix}$$

where v_i is a node in a constructed graph. X_D and X_{Dp} are the embeddings of the reference dataset and their corresponding adversarial examples, respectively. The newly generated adversarial example I_{adv} is pushed to be far away from the adversarial examples of the generated graph at each iteration. The intuition is that, ideally, a graph consisting only of benign examples is more likely to be classified as benign, and this requires moving the white-box adversary towards the decision boundaries of an adversary class, while moving away from the possible nearby adversarial examples. This process requires regenerating the graph of I_{adv} in each iteration of the attack, since the applied perturbation affects the embedding space. We refer to this attack as CW_{wb}.

Figure 4 shows the t-distributed stochastic neighbor embeddings (t-SNE) visualization [45] of the embedding space of CIFAR-10 benign and adversarial examples. Note that the benign examples are grouped in ten different clusters, each corresponding to a unique class. The white-box CW_{wb} attack creates adversarial examples that are very close to the benign clusters. Such examples are significantly difficult to detect, given their minimal (visual) perturbation.

Gray-box Setting. In this setting, the adversary is unaware of the deployed adversarial defense, but knows the pre-trained classifier's parameters. For the decision boundary attack, however, only an oracle to query the classifier for the prediction output is provided to the adversary. Unless stated otherwise, the threat model is assumed to be gray-box (*i.e.* unaware of the implemented defense).

In the experiments, we focus on the detection of adversarial examples with relatively low perturbation. In particular, we consider the following parameters for the adversarial attack: PGD: $\delta = 0.02$, step size of 0.002 with 50 iterations, FGSM: $\delta = 0.05$, CW: $\delta = 0.10$, learning rate of 0.03,

Table 3: The (AUC) of different adversarial detection approaches. Left: the performance of different adversarial detection approaches. LID and ours are trained on the same attack evaluated on. Right: the LID and ours are trained on CW adversarial examples, and tested on different unseen attacks. [1] Due to memory and resources constraints, AutoAttack and Square adversarial examples are only generated for CIFAR-10 and STL-10 datasets.

Dataset	Approach	FGSM	PGD	CW	AutoAttack	Square	Boundary	CW_{wb}	FGSM	PGD	AutoAttack	Square	Boundary	CW_{wb}
CIFAR-10	DkNN [47]	61.50%	51.18%	61.46%	52.11%	59.51%	70.11%	60.37%	61.50%	51.18%	52.11%	59.51%	70.11%	60.37%
	kNN [4]	61.80%	54.46%	65.25%	52.64%	73.39%	75.88%	59.75%	61.80%	54.46%	52.64%	73.39%	75.88%	59.75%
	LID [38]	73.56%	67.95%	55.60%	56.25%	85.93%	99.48%	55.28%	71.15%	61.27%	55.57%	66.11%	97.01%	55.28%
	Hu [33]	84.44%	58.55%	**90.99%**	53.54%	95.83%	90.71%	78.33%	84.44%	58.55%	53.54%	**95.83%**	90.71%	78.33%
	LNG	**99.88%**	**91.39%**	89.74%	**84.03%**	**98.82%**	**99.98%**	**84.38%**	**98.51%**	**63.14%**	**58.47%**	94.71%	**99.92%**	**84.38%**
ImageNet[1]	DkNN [47]	89.20%	78.00%	68.80%	—	—	76.80%	68.80%	89.20%	78.00%	—	—	76.80%	68.80%
	kNN [4]	51.60%	51.10%	50.70%	—	—	56.90%	50.50%	51.60%	51.10%	—	—	56.90%	50.50%
	LID [38]	99.26%	98.14%	58.75%	—	—	**100%**	57.76%	90.58%	52.45%	—	—	96.16%	57.76%
	Hu [33]	72.59%	86.00%	80.82%	—	—	63.20%	80.44%	72.59%	86.00%	—	—	63.20%	80.44%
	LNG	**99.53%**	**98.42%**	**86.05%**	—	—	**100%**	**86.49%**	**96.85%**	**89.61%**	—	—	**99.93%**	**86.49%**
STL-10	DkNN [47]	60.66%	59.33%	57.49%	60.77%	50.10%	62.93%	62.00%	60.66%	59.33%	60.77%	50.10%	62.93%	62.00%
	kNN [4]	59.40%	58.60%	65.40%	55.27%	65.15%	59.80%	58.43%	59.40%	58.60%	55.27%	65.15%	59.80%	58.43%
	LID [38]	80.84%	74.12%	60.87%	73.44%	73.78%	99.86%	60.26%	69.59%	56.06%	55.80%	62.98%	**100%**	60.26%
	Hu [33]	57.86%	86.45%	81.07%	64.01%	80.64%	59.74%	63.33%	57.86%	**86.45%**	64.01%	80.64%	59.74%	63.33%
	LNG	**99.40%**	**99.35%**	**93.95%**	**99.37%**	**82.20%**	**100%**	**91.13%**	**88.08%**	69.20%	**68.49%**	**90.32%**	**100%**	**91.13%**

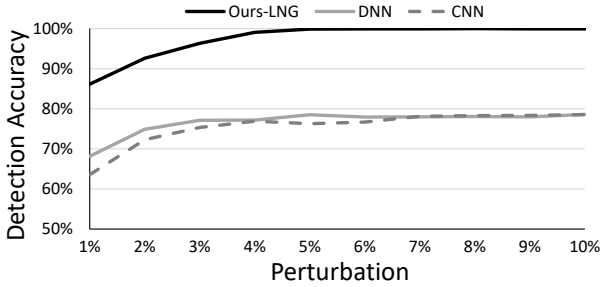
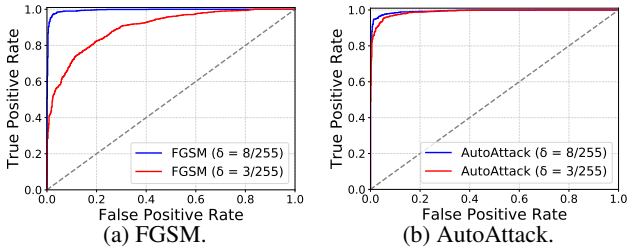


Figure 5: The AUC (%) performance of the GNNs-based discriminator compared to the convolutional and deep neural networks-based discriminators for different FGSM perturbation rates ($\delta \in [0.01,0.1]$), using the LNG-based graph on CIFAR-10 dataset.



(a) FGSM. (b) AutoAttack.

Figure 6: The ROC-AUC of FGSM and AutoAttack on STL-10 dataset using perturbation size (δ) of $\frac{8}{255}$ and $\frac{3}{255}$.

and 50 iterations, AutoAttack: $\delta = \frac{8}{255}$, Square: $\delta = \frac{8}{255}$, Boundary: $\delta = 0.10$ with 100 iterations. We select the smallest perturbation (δ) that still achieves more than 50% attack success rate on the original model. While adversarial examples generated with low perturbation have a lower attack success rate, their detection is significantly challenging. Figure 5 shows the performance of the graph-based discriminator, trained on k-NNG generated graphs, in detecting FGSM-generated adversarial examples for different perturbation rates (δ). The proposed approach is benchmarked against a convolutional neural network (CNN)-based discriminator trained using adversarial training, and a fully-connected layer-based discriminator (DNN) that directly works on the input image and its embedding [39]. Both DNN and CNN-based methods converge to the same point because they use the same embedding information. Note that their performance is worse than LNG because they do not use dynamic relations in the embedding neighborhood.

4.3. Comparison with state-of-the-art

Detecting known attacks: Table 3 (left) compares the performance of the proposed method on detecting adversarial examples generated using known attacks with four state-of-the-art adversarial example detection approaches: DkNN [47], kNN [4], LID [38] and Hu *et al*. [33], on three datasets, CIFAR-10, STL-10, and ImageNet. The results are reported using area under the ROC curve metric (AUC). The LID and the proposed detection method are trained and tested using the same adversarial attack methods, except for CW_{wb} attack, where the detector is trained on the traditional CW attack. We report the performance of the graph discriminator trained using the Latent Neighborhood Graph (LNG) and the k-Nearest-Neighbor Graph (k-NNG). Experimental results demonstrate that the proposed approach outperforms state-of-the-art adversarial example detection methods on both datasets. The performance benefit is especially significant in the detection of white-box (CW_{wb}) attack, where adversarial and benign space are deeply interleaved with each other (as illustrated in Figure 4). This is because our algorithm generates a highly discriminative neighborhood graph based on the input example's local manifold structure, and hence it is able to distinguish between adversarial

Table 4: The (AUC) performance (%) of our approach using clean vs. adversarially augmented (Adv.) *reference* sets.

Dataset	Approach	Adv.	FGSM	PGD	CW	Boundary	CW_{wb}
CIFAR-10	k-NNG	×	97.38	54.29	86.45	99.92	74.84
		✓	99.54	85.78	89.63	99.89	81.44
	LNG	×	99.24	85.62	**89.91**	99.96	80.77
		✓	**99.88**	**91.39**	89.74	**99.98**	**84.38**
ImageNet	k-NNG	×	97.25	90.78	50.49	99.91	51.56
		✓	**99.58**	98.36	79.03	**100**	77.01
	LNG	×	99.40	94.98	81.24	99.99	81.64
		✓	99.53	**98.42**	**86.05**	**100**	**86.49**

and benign examples with high accuracy. Similar performance benefit is observed in case of FGSM and Autoattacks on STL-10 dataset (see **Figure 6**).

Detecting unseen attacks: In this experiment, we compare the robustness of the proposed method against unseen adversarial attacks to state-of-the-art adversarial detection methods. Each adversarial example detection method is trained using CW attack, and evaluated on other attacks. The results are shown in Table 3 (right). The proposed approach outperforms other methods by a significant margin on different attack configurations.

4.4. Ablation Study

The objective of this experiment is to compare the performance of k-NNG and LNG with and without using adversarial examples from the reference dataset. The results are shown in Table 4 for CIFAR-10 and ImageNet. The edge estimation process used to construct the LNG improves the overall performance of the proposed detection method. Significant performance improvement is also observed when using reference adversarial examples as it results in better estimation of the neighborhood of the input image. The reported improvement due to the use of adversarial examples (over 20% in some cases) is especially beneficial in detecting stronger attacks (PGD, and CW).

4.5. Impact of Graph Topology

The objective of this experiment is to investigate the impact of graph topology on detection performance. The following graph types are compared: i) the original k-nearest neighbor-based graph (k-NNG, as described in section 3.2.1), ii) graph with no connections between nodes (NC), iii) graph with connections between all nodes (AC), iv) the k-NNG where the center node is connected to all nodes in the neighborhood (CC), and v) the proposed latent neighborhood graph (LNG) where the input node is connected to all nodes with estimated edges between the neighborhood nodes. Table 5 presents the performance of the detector trained on each graph for CIFAR-10, and ImageNet datasets, where the discriminator is trained and evaluated on the same attack configuration. Overall, connecting the center node with neighbor nodes helped aggregate the neighborhood information towards the input example, which improves the performance. By connecting the neighborhood nodes adaptively, LNG provides better context for graph discriminator.

Table 5: The (AUC) performance (%) of using different connections configurations in the neighborhood graph. NC: no connections between nodes, AC: all connected graph, CC: only the center node is connected to all nodes.

Dataset	Approach	FGSM	PGD	CW	Boundary	CW_{wb}
CIFAR-10	k-NNG	99.54	85.78	89.63	99.89	81.44
	NC	99.72	87.21	87.53	99.81	81.60
	AC	99.83	87.72	90.67	99.83	80.43
	CC	99.72	88.67	**91.51**	99.94	82.92
	LNG	**99.88**	**91.39**	89.74	**99.98**	**84.38**
ImageNet	k-NNG	99.58	98.36	79.03	**100**	77.01
	NC	**99.70**	**98.51**	77.42	**100**	74.99
	AC	99.66	98.35	79.74	**100**	78.15
	CC	99.66	98.23	78.19	**100**	75.59
	LNG	99.53	98.42	**86.05**	**100**	**86.49**

4.6. Graph Detection: Time Comparison

For LNG method, on average, the detection process of each image takes 1.55 and 1.53 seconds for CIFAR-10 and ImageNet datasets, respectively. The time includes (i) embedding extraction, (ii) neighborhood retrieval, (iii) LNG construction, and (iv) graph detection. This is significantly lower in comparison to Hu *et al*. [33], which requires an average of 14.05 and 5.66 seconds to extract the combined characteristics from CIFAR-10 and ImageNet dataset, respectively.

5. Conclusion

Detection of adversarial examples, particularly generated using unseen adversarial attacks, is a challenging security problem for deployed deep neural network classifiers. In this work, we propose the first graph-based adversarial example detection method that generates latent neighborhood graphs in the embedding space of a pre-trained classifier to detect adversarial examples. The proposed method achieves state-of-the-art adversarial example detection performance against various white- and gray-box adversarial attacks on three benchmark datasets. We also show the effectiveness of the proposed approach on unseen attacks, where training our approach using a strong adversarial attack (CW) enables robust detection of adversarial examples generated using other attacks.

Acknowledgement. The work of D. Mohaisen and the validating and reporting work of this publication by A. Abusnaina was supported in part by NRF under the 2016 grant 2016K1A1A2912757. The system design was done when A. Abusnaina was an intern at Visa research.

References

[1] N. Carlini and D. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

[2] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy*, 2017.

[3] Adam Coates *et al*. An analysis of single-layer networks in unsupervised feature learning. In *The international conference on artificial intelligence and statistics*, pages 215–223, 2011.

[4] A. Dubey *et al*. Defense against adversarial images using web-scale nearest-neighbor search. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2019.

[5] A. Krizhevsky *et al*. Learning multiple layers of features from tiny images. 2009.

[6] A. Krizhevsky *et al*. Imagenet classification with deep convolutional neural networks. In *26th Annual Conference on Neural Information Processing Systems*, 2012.

[7] A. Kurakin *et al*. Adversarial machine learning at scale. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[8] A. M. Nguyen *et al*. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2015.

[9] Angelo Sotgiu *et al*. Deep neural rejection against adversarial examples. *EURASIP J. Inf. Secur.*, 2020:5, 2020.

[10] A. Voulodimos *et al*. Deep learning for computer vision: A brief review. *Comput. Intell. Neurosci.*, 2018:7068349:1–7068349:13, 2018.

[11] C. Szegedy *et al*. Intriguing properties of neural networks. In *2nd International Conference on Learning Representations, ICLR*, 2014.

[12] D. Bahdanau *et al*. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[13] F. Schroff *et al*. Facenet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, pages 815–823. IEEE Computer Society, 2015.

[14] F. Tramèr *et al*. Ensemble adversarial training: Attacks and defenses. *CoRR*, 2017.

[15] G. Cohen *et al*. Detecting adversarial samples using influence functions and nearest neighbors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

[16] G. Hinton *et al*. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal processing magazine*, 29(6):82–97, 2012.

[17] G. Huang *et al*. Densely connected convolutional networks. In *IEEE conference on computer vision and pattern recognition*, 2017.

[18] Hasan Ferit Eniser *et al*. RAID: randomized adversarial-input detection for neural networks. *CoRR*, abs/2002.02776, 2020.

[19] I. J. Goodfellow *et al*. Explaining and harnessing adversarial examples. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[20] J.H. Metzen *et al*. On detecting adversarial perturbations. In *5th International Conference on Learning Representations, ICLR*, 2017.

[21] J. Svoboda *et al*. Peernets: Exploiting peer wisdom against adversarial attacks. *arXiv preprint arXiv:1806.00088*, 2018.

[22] J. Tenenbaum *et al*. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[23] K. Lee *et al*. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 7167–7177, 2018.

[24] K. Simonyan *et al*. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR*, 2015.

[25] Luca Cosmo *et al*. Latent patient network learning for automatic diagnosis. *arXiv preprint arXiv:2003.13620*, 2020.

[26] L. Engstrom *et al*. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, 2017.

[27] M. Nicolae *et al*. Adversarial robustness toolbox v1. 0.0. *arXiv preprint arXiv:1807.01069*, 2018.

[28] M. Takeru *et al*. Distributional smoothing with virtual adversarial training. *arXiv preprint arXiv:1507.00677*, 2015.

[29] N. Papernot *et al*. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symposium on Security and Privacy, SP*, 2016.

[30] O. Russakovsky *et al*. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.

[31] P. Velickovic *et al*. Graph attention networks. *CoRR*, 2017.

[32] R. Feinman *et al*. Detecting adversarial samples from artifacts. *CoRR*, abs/1703.00410, 2017.

[33] S. Hu *et al*. A new defense against adversarial images: Turning a weakness into a strength. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS*, pages 1633–1644, 2019.

[34] Shiqing Ma *et al*. NIC: detecting adversarial samples with neural network invariant checking. In *26th Annual Network and Distributed System Security Symposium, NDSS*. The Internet Society, 2019.

[35] U. Shaham *et al*. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 2018.

[36] W. Xu *et al*. Feature squeezing: Detecting adversarial examples in deep neural networks. In *25th Annual Network and Distributed System Security Symposium, NDSS*, 2018.

[37] W. Zhang *et al*. Towards end-to-end speech recognition with deep multipath convolutional neural networks. In *12th International Conference on Intelligent Robotics and Applications ICIRA*, 2019.

[38] X. Ma *et al*. Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

[39] Z. Gong *et al*. Adversarial and clean data are not twins. *CoRR*, abs/1704.04960, 2017.

[40] K. He*et al*. Deep residual learning for image recognition. In *IEEE conference on computer vision and pattern recognition*, 2016.

[41] D. Hendrycks and T. G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *arXiv preprint arXiv:1807.01697*, 2018.

[42] D. Jakubovitz and R. Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 514–529, 2018.

[43] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP*, pages 1746–1751. ACL, 2014.

[44] X. Li and F. Li. Adversarial examples detection in deep networks with convolutional filter statistics. In *IEEE International Conference on Computer Vision, ICCV*, 2017.

[45] L.V.D. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 2008.

[46] D. Meng and H. Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS*, pages 135–147. ACM, 2017.

[47] N. Papernot and P. D. McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *CoRR*, abs/1803.04765, 2018.

[48] C. Sitawarin and D. Wagner. Defending against adversarial examples with k-nearest neighbor. *arXiv preprint arXiv:1906.09525*, 2019.

[49] X. Yuan*et al*. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.