IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 16, 2019, accepted August 15, 2019, date of publication September 4, 2019, date of current version September 17, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2939352

PuVAE: A Variational Autoencoder to Purify Adversarial Examples

UIWON HWANG[1], JAEWOO PARK[2], HYEMI JANG[1], SUNGROH YOON[1], (Senior Member, IEEE), AND NAM IK CHO[2], (Senior Member, IEEE)

[1]Electrical and Computer Engineering, Seoul National University, Seoul 08826, South Korea
[2]Department of Electrical and Computer Engineering, INMC, Seoul National University, Seoul 08826, South Korea

Corresponding authors: Sungroh Yoon (sryoon@snu.ac.kr) and Nam Ik Cho (nicho@snu.ac.kr)

This work was supported in part by the Projects for Research and Development of Police Science and Technology through the Center for Research and Development of Police Science and Technology and the Korean National Police Agency funded by the Ministry of Science, ICT and Future Planning under Grant PA-C000001, in part by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government (Ministry of Science and ICT) under Grant 2018R1A2B3001628, in part by the Samsung Electronics (DS and Foundry), and in part by the Brain Korea 21 Plus Project in 2019.

ABSTRACT Deep neural networks are widely used and exhibit excellent performance in many areas. However, they are vulnerable to adversarial attacks that compromise networks at inference time by applying elaborately designed perturbations to input data. Although several defense methods have been proposed to address specific attacks, other types of attacks can circumvent these defense mechanisms. Therefore, we propose Purifying Variational AutoEncoder (PuVAE), a method to purify adversarial examples. The proposed method eliminates an adversarial perturbation by projecting an adversarial example on the manifold of each class and determining the closest projection as a purified sample. We experimentally illustrate the robustness of PuVAE against various attack methods without any prior knowledge about the attacks. In our experiments, the proposed method exhibits performances that are competitive with state-of-the-art defense methods, and the inference time is approximately 130 times faster than that of Defense-GAN which is a state-of-the art purifier method.

INDEX TERMS Adversarial attack, variational autoencoder, deep learning.

I. INTRODUCTION

Significant developments in deep learning has led to its use in several areas including image recognition [1]–[4], disease prediction [5], and autonomous driving [6]. However, security issues in deep neural networks, especially vulnerability to adversarial attacks, are emerging [7], [8]. The goal of adversarial attacks is to fool a target deep neural network via applying elaborately designed perturbation to input data [7]. Adversarial attacks make real-world application of deep neural networks hazardous. In autonomous driving [9], such attacks can cause an accident by tricking an object detector to recognize pedestrians as roads.

Adversarial attacks can be classified into white-box attacks and black-box attacks [10]. In white-box attacks, an attacker can access both the model parameters and training dataset. She/he uses the information to perturb test data so that the target classifier predicts the adversarial example into one of other classes different from the ground truth. In black-box attacks, an attacker cannot obtain the model parameters. Therefore, several samples need to be fed to the target classifier to obtain the predicted labels [11]. Black-box attacks create attack samples by applying a white-box method to a trained substitute model with labels inferred from the target classifier. This mechanism is derived from the fact that adversarial examples that successfully trick a model can deceive other models as well [12]. The target model is attacked based on the transferability of adversarial examples.

To address these attacks, several defense mechanisms have been proposed. There are three categories of defense mechanisms. The first category involves modifying the training dataset such that the target classifier is robust against adversarial attacks [7], [8], [13], [14]. This method is easy to apply but blocks only the type of attacks used while training.

The associate editor coordinating the review of this article and approving it for publication was Oguz Elibol.

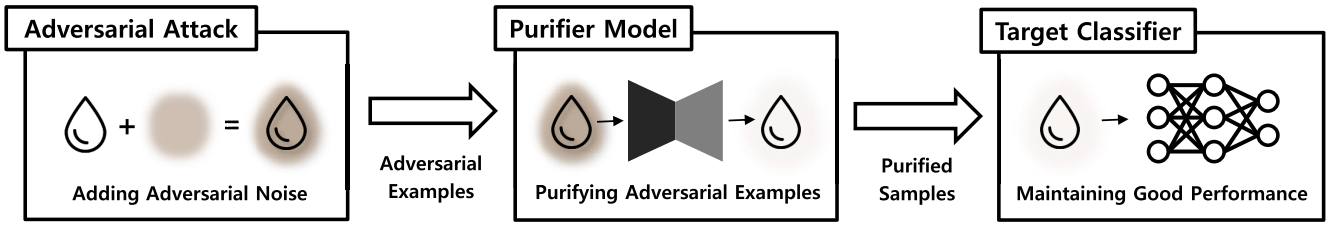


FIGURE 1. Overview of the defense mechanisms using a purifier model. An adversarial example is generated by adding an adversarial perturbation to an input sample. A purifier model is adopted to defend the adversarial attack by removing the adversarial perturbation from the adversarial example. The purified sample is delivered to the target classifier, and the target classifier predicts the correct label from the cleaned sample.

The second category blocks gradient calculation via changing the training procedure [15]–[18]. However, this mechanism is effective only for gradient-based attacks. The third category involves removing adversarial noises from the samples fed to the target classifier [19]–[21].

Our main focus of defense mechanisms is the purification of input data that may have adversarial perturbations. These mechanisms can effectively address adversarial examples regardless of the attack methods. Purifying methods use a generative model to learn the data distribution and project the adversarial example onto the learned data distribution $p(x)$. The generative models are called *purifiers*. Recently, MagNet [20] and Defense-GAN [19] were proposed. MagNet uses autoencoders to suggest fast and simple method to defend adversarial attacks. However, MagNet produces unstable performances depending on the datasets and attack methods. Our experiments show that Defense-GAN using a generative adversarial network (GAN) has a better defense performance than MagNet. However, Defense-GAN takes a lot of time to purify adversarial examples. Figure 1 shows the overview of defense mechanisms using a purifier model.

In this paper, we aim to rapidly generate well-classified samples from adversarial examples. The *purified* samples are fed to the target classifier so that they can be classified without being affected by adversarial perturbations. To solve the limitations of MagNet and Defense-GAN, we propose Purifying Variational AutoEncoder (PuVAE), which purifies adversarial examples using a variational autoencoder (VAE). The proposed method uses variational inference to generate samples, and provides comparable or better defense performance than that of state-of-the-art methods. In contrast to Defense-GAN, PuVAE generates clean samples with one feed-forward step. Therefore, our method is robust against adversarial attacks within a reasonable time limit.

In summary, our contributions in this paper are as follows:

- We propose a VAE-based defense method, PuVAE, to effectively purify adversarial attacks. The proposed method shows a remarkable performance over other defense methods.
- The proposed method significantly reduces the time to generate purified samples. Within a reasonable time limit, PuVAE outperforms state-of-the-art defense methods.
- Experimental results demonstrate that the proposed method functions robustly against a variety of attack methods and datasets.

We explain various adversarial attacks and defense methods followed by comparison with our proposed method in Section II and explain the motivation of PuVAE in Section III. The mechanism of PuVAE and the experimental results are explained in Section IV and Section V, respectively. Finally, we conclude our findings in Section VI.

II. ADVERSARIAL ATTACK AND DEFENSE

A. ADVERSARIAL ATTACK METHODS

An adversarial example is a sample that is misclassified by the target classifier by using an intended noise that is not perceivable by humans. Various adversarial attacks for deep neural networks have been proposed since the work of Szegedy *et al.* [7]. Depending on the intention of an attacker, attacks are categorized as untargeted and targeted attacks. Untargeted attacks make a model misclassify an input to other class except the true label. Targeted attacks aim to fake a model to classify an input to a specific class as chosen by the attacker. As defending targeted attacks is difficult than defending untargeted attacks [22], we use targeted attacks for our study.

Goodfellow *et al.* [8] claimed that the cause of vulnerability to adversarial examples is a linear characteristic of neural networks and proposed the fast gradient sign method (FGSM) that uses the gradient of the objective function of neural networks. As FGSM uses simple operations compared to the previous method [7] and its linearity assumption is easy to analyze, various attacks based on FGSM have been proposed [13], [23], [24]. Among them, we use targeted FGSM [23], which is a one-step gradient method. The mechanism to generate adversarial examples is presented as follows:

$$\hat{\mathbf{x}} = \mathbf{x} - \varepsilon \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\mathbf{x}, y_{\text{target}}; \theta)), \tag{1}$$

where $\hat{\mathbf{x}}$ is the adversarial example of targeted FGSM, $\mathbf{x}$ denotes the original sample, ε is the perturbation size, $\mathcal{L}$ is the objective function of the target classifier, θ is the parameters of the target classifier, and y_{target} is the selected label that is randomly chosen among classes except the true label of $\mathbf{x}$. Although FGSM is a fast algorithm, it is easy to defend the one-step gradient-based approach.

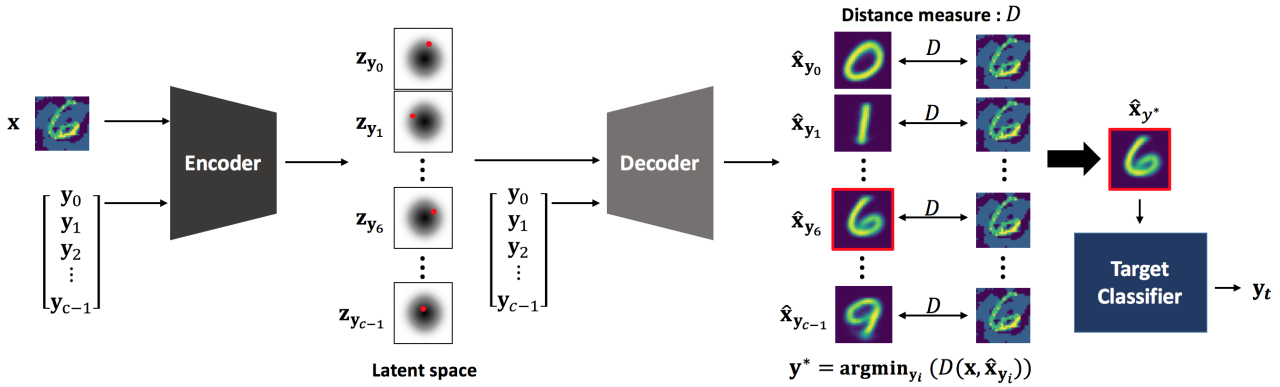


FIGURE 2. Inference of PuVAE with an MNIST image; A latent vector $\mathbf{z}_{\mathbf{y}_i}$ is sampled from an adversarial sample $\mathbf{x}$ and a class condition $\mathbf{y}_i$. $i \in [0, c-1]$ is a class index where c is the number of classes. $\hat{\mathbf{x}}_{\mathbf{y}_i}$ denotes a candidate for the purified image. The purified image $\hat{\mathbf{x}}_{\mathbf{y}^*}$ with a minimum distance from $\mathbf{x}$ enters the target classifier.

To overcome this problem, iterative methods [25], [26] were proposed to optimize an adversarial noise in several steps with a small perturbation allowing a more sophisticated attack. For our study, we use the targeted-version of iterative FGSM (targeted iFGSM) [25], and the perturbation is generated as follows:

$$\begin{aligned} \hat{\mathbf{x}}_0 &= \mathbf{x}, \\ \hat{\mathbf{x}}_{i+1} &= \text{Clip}_{\mathbf{x},\varepsilon}\{\hat{\mathbf{x}}_i - \alpha \cdot \text{sign}(\nabla_{\mathbf{x}} \mathcal{L}(\hat{\mathbf{x}}_i, y_{\text{target}}; \theta))\}, \end{aligned} \quad (2)$$

where $\hat{\mathbf{x}}_i$ is the intermediate result of the targeted adversarial example in i-th iteration, $\mathbf{x}$ is the original image, ε denotes the possible perturbation range, α is the size of small perturbation at each step, $\mathcal{L}$ denotes the objective function of the target classifier, θ is the parameters of the target classifier, and y_{target} is a randomly selected class except the true label of $\mathbf{x}$. Targeted iFGSM shows powerful attack performances with a small perturbation size than that of targeted FGSM.

In addition to attacks based on FGSM, convex optimization based methods are also widely used to create an adversarial perturbation [7], [27]. The Carlini and Wagner (CW) attack [27] defeats defensive distillation [28] and conveys that it is the most powerful attack method among all the existing methods. CW attack is quasi-imperceptible using restriction of l_0, l_2 and l_∞ norm constraints. The adversarial perturbation is derived as follows:

$$\begin{aligned} &\text{minimize } ||\delta||_p + c \cdot f(\mathbf{x} + \delta) \\ &\text{subject to } \mathbf{x} + \delta \in [0, 1]^N, \end{aligned} \quad (3)$$

where $f(\mathbf{x})$ denotes the function that aims to lead model misclassification, δ denotes the perturbation added to the input image $\mathbf{x}$, N is the input dimension, and c is a positive scalar value that is found empirically. In this study, we use default p and c values from open source software CleverHans[1] by Papernot *et al.* [29] to verify whether our proposed method can defend the CW attack. Further details on solving Equation 3 are presented in [27].

[1]https://github.com/tensorflow/cleverhans

To compare the defense performances, we use three attack mechanisms, targeted FGSM, targeted iFGSM and the CW attack. For convenience of notation, we dub targeted FGSM and iFGSM as FGSM and iFGSM, respectively, throughout this paper.

B. DEFENSE METHODS

Defense methods can be categorized into three types. The first type modifies training data to regularize a target classifier. The second type blocks gradients calculations of the target classifier. The third type purifies inputs that enter the target classifier. In this paper, we use the regularization method and purifying method as baselines to compare the performance of PuVAE.

Regularization methods are easy to apply, and adversarial training [8] is one of the most famous regularization methods. Adversarial training utilizes training data mixed with adversarial attacks to train a target classifier. The objective function of adversarial training is as follows:

$$\begin{aligned} \hat{\mathcal{L}}(\mathbf{x}, y; \theta) = \alpha \mathcal{L}(\mathbf{x}, y; \theta) + (1 - \alpha) \mathcal{L} \\ \times (\mathbf{x} + \varepsilon \cdot \text{sign}(\nabla_x \mathcal{L}(\mathbf{x}, y; \theta)), y; \theta), \end{aligned} \quad (4)$$

where y is the true label of the input $\mathbf{x}$. $\mathcal{L}$ is the objective function (e.g., cross-entropy function) of the target classifier and α is the hyperparameter tuning the ratio between normal objective function and adversarial objective function based on FGSM. θ denotes the parameters of the target classifier. The method successfully defends the attacks similar to the ones used while training, but fails to defend other kinds of attacks.

However, purifying methods are not concerned with the types of attacks because training data is only used to train a purifier for defending purpose. The methods use a generative model to generate cleaned samples from attack samples. Specifically, MagNet [20] learns the distribution of original data using one or more autoencoders called the reformer networks and trained using reconstruction loss as follows:

$$\mathcal{L}(\mathbf{x}; \theta) = ||\mathbf{x} - \hat{\mathbf{x}}||_2^2, \quad (5)$$

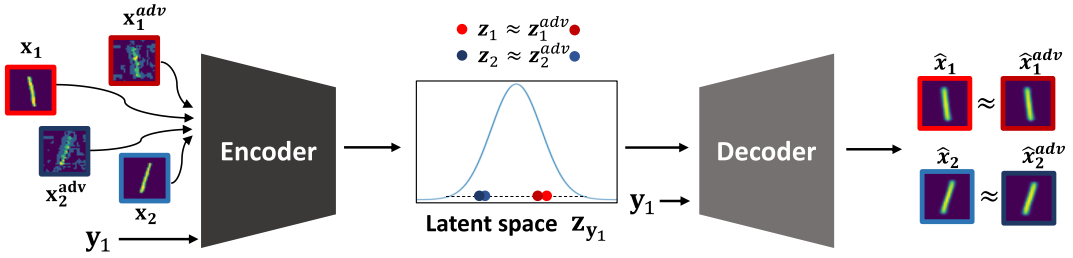


FIGURE 3. Illustration of properties of VAE related to adversarial attacks. Each pair of an original sample and adversarial example generated from it is represented by similar colored boxes. The images that share the same geometric properties are projected to contiguous areas in the latent space. The decoder generates analogous images from latent vectors that have similar values.

where $\mathbf{x}$ denotes the input image and $\hat{\mathbf{x}}$ denotes the reconstructed image by the autoencoders. θ represents the parameters of the autoencoders. At inference time, MagNet passes an input data to the autoencoders that move the input data closer to the training data manifold and purified data are supplied to the classifier. However, the method exhibits poor performances when compared with that of Defense-GAN.

Defense-GAN [19] uses the characteristics of GANs to defend a target model against adversarial attacks. It uses the fact that optimizing the objective function of a GAN is equivalent to making the generator distribution p_g identical to the data distribution p_{data}. After training the GAN using the original data, Defense-GAN finds the latent vector $\mathbf{z}$ to minimize the reconstruction error between the generated sample $G(\mathbf{z})$ and the input sample $\mathbf{x}$ that might have an adversarial noise as follows:

$$\min_{\mathbf{z}} ||G(\mathbf{z}) - \mathbf{x}||_2^2. \tag{6}$$

The reconstruction error is minimized by iteratively applying gradients of the generator to $\mathbf{z}$. Subsequently, data generated with optimal $\mathbf{z}$ are supplied to the target classifier as an input. Defense-GAN relies on the unstable performance of GAN, which occationally reproduces the adversarial noise by directly optimizing errors between the input data $\mathbf{x}$ and the generated sample $G(\mathbf{z})$. In addition, it takes a long time to yield maximum defense performance due to the iterative nature of Defense-GAN. Hence, for real-time applications such as object detection, which must operate quickly, a fast defense algorithm needs to be developed.

III. PROPERTIES OF VARIATIONAL AUTOENCODER RELATED TO ADVERSARIAL ATTACK

Bengio and Vincent [30] used a generative model to represent data distribution. The relation in most data are too complex to be directly discovered, thus relatively simple latent variables are used to represent data distribution. Kingma and Welling [31] introduced VAE, which is a method that uses a combination of neural networks and variational inference. VAE consists of an encoder and a decoder. The encoder receives data as the input and produces outputs, which are the mean and standard deviation of the latent vector distribution. The decoder receives the latent variable, which is sampled from the latent vector distribution, and uses it to reconstruct the input data. The objective function of VAE is given as follows:

$$\log(p_\theta(\mathbf{x})) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \mathcal{L}(\mathbf{x};\theta,\phi), \tag{7}$$

$$\mathcal{L}(\mathbf{x};\theta,\phi) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log(p_\theta(\mathbf{x}|\mathbf{z}))], \tag{8}$$

where $\log(p_\theta(\mathbf{x}))$ denotes the marginal log-likelihood of the data, $\mathcal{L}(\mathbf{x};\theta,\phi)$ denotes the variational lower bound of marginal likelihood, $p_\theta(\mathbf{x}|\mathbf{z})$ denotes the output distribution of the decoder, $q_\phi(\mathbf{z}|\mathbf{x})$ denotes the output distribution of the encoder, $p_\theta(\mathbf{z})$ denotes the normal distribution, and $p_\theta(\mathbf{z}|\mathbf{x})$ is the true posterior. Because $p_\theta(\mathbf{z}|\mathbf{x})$ is intractable [31], maximizing the lower bound $\mathcal{L}(\mathbf{x};\theta,\phi)$ is generally used to maximize marginal likelihood of data.

Sohn *et al.* [32] indicated that a conditional VAE (cVAE) is specifically used to learn a multimodal distribution via additional class information. The basic idea of cVAE is similar to that of VAE except that cVAE aims to learn data distribution for each class. Therefore, both of the encoder and decoder of cVAE take a class label as an additional input. The objective function of cVAE is represented as follows:

$$\log(p_\theta(\mathbf{x}|\mathbf{y})) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z}|\mathbf{x},\mathbf{y})) + \mathcal{L}(\mathbf{x},\mathbf{y};\theta,\phi), \tag{9}$$

$$\mathcal{L}(\mathbf{x},\mathbf{y};\theta,\phi) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})||p_\theta(\mathbf{z})) + \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})}[\log(p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y}))], \tag{10}$$

where $\mathbf{y}$ denotes the class label of an input $\mathbf{x}$, $\log(p_\theta(\mathbf{x}|\mathbf{y}))$ denotes the conditional marginal log-likelihood of the data, $\mathcal{L}(\mathbf{x},\mathbf{y};\theta,\phi)$ denotes the variational lower bound of conditional marginal likelihood, $p_\theta(\mathbf{x}|\mathbf{z},\mathbf{y})$ denotes the conditional output distribution of the decoder, $q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y})$ denotes the conditional output distribution of the encoder, $p_\theta(\mathbf{z})$ denotes the normal distribution, and $p_\theta(\mathbf{z}|\mathbf{x},\mathbf{y})$ is conditional true posterior. Since $p_\theta(\mathbf{z}|\mathbf{x},\mathbf{y})$ is intractable, maximizing the lower bound $\mathcal{L}(\mathbf{x},\mathbf{y};\theta,\phi)$ is generally used to maximize the marginal likelihood of data.

Figure 3 describes the properties of VAE related to adversarial attacks. The original goal of cVAE is to get the decoder that generates class conditional data following real data distribution. Although the encoder has been only used for learning purposes, its characteristics could also be suitable for erasing adversarial perturbations.

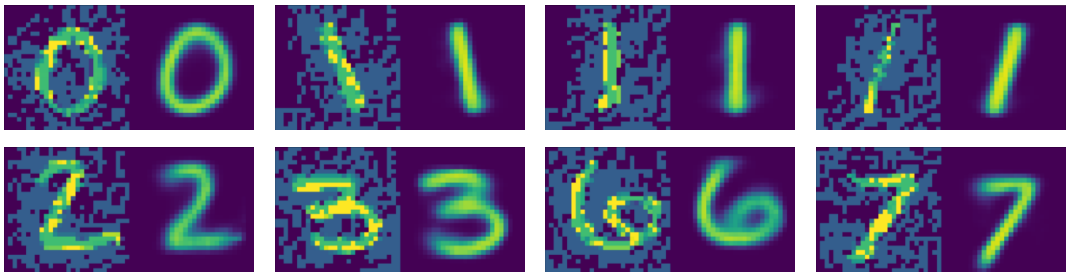


FIGURE 4. Examples of purified images from PuVAE on the MNIST dataset. In each image, the left-hand side shows the image attacked with FGSM ($\epsilon = 0.3$), and the right-hand side shows the purified result of the left-hand side image using PuVAE.

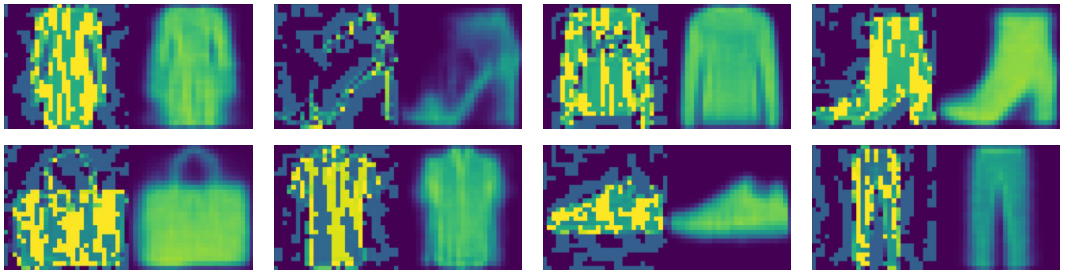


FIGURE 5. Examples of purified images from PuVAE using the Fashion-MNIST dataset. In each image, the left-hand side shows the image attacked with FGSM ($\epsilon = 0.3$), and the right-hand side shows the purified result of the left-hand side image using PuVAE.

Maximizing $-\mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{y})||p_\theta(\mathbf{z}))$ forces the latent vector $\mathbf{z}$ extracted by the encoder to present the normal distribution. We postulate that not just the diverse training samples but also the adversarial examples are mapped to the normal latent space.

Zhu *et al.* [33] showed that a cyclic structure (e.g., A $\rightarrow$ B $\rightarrow$ A), with the use of reconstruction loss, learns to change the detail or low-level texture while relatively maintaining the geometric characteristics in the translated domain. cVAE uses the cyclic reconstruction loss. Therefore, we expect the encoder to be robust against the perturbation of low level features including an adversarial noise. We also expect the encoder to map the input image and adversarial sample, created using the input image, to nearby locations while maintaining geometric features.

Kingma and Welling [31] visualized generated samples from latent vectors at similar positions in latent space, and showed that the generated samples have the same morphological attributes. From this observation, the decoder can generate images that have the same morphological attributes from contiguous latent vectors.

Finally, using the above two losses (the Kullback-Leibler divergence and the reconstruction loss), we anticipate that the encoder outputs similar latent vectors in normal distribution regardless of adversarial perturbations. That is, the encoder is a function that maps the clean image and the image with adversarial perturbation very closely in latent space. In addition, since the decoder has only learned the clean samples in the learning process, the decoder works as a generator to output a sample on the manifold of the clean images. Through the entire encoder-decoder process, we deduce that an image similar to the clean image is generated when a clean image is entered. On the other hand, when an adversarial example is entered, the adversarial perturbation is removed in the encoder and the purified sample is generated in the decoder.

We analyze the purified images on the MNIST and Fashion-MNIST datasets through experiments. Figures 4 and 5 are the visualizations of the experimental results on the MNIST and Fashion-MNIST datasets, respectively. Particularly, the PuVAE purifies adversarial examples while maintaining the angle of number 1 in the first row of Figure 4. Therefore, we confirm that cVAE selectively removes the adversarial noise while preserving the original shape and position of the image as we expect.

Based on the above reasons, we use cVAE as the base model for our proposed method. We confirm that the forwarding process via the encoder-decoder model effectively purifies adversarial noises from data.

IV. PROPOSED METHOD

In this paper, we propose a VAE-based defense method called PuVAE to purify adversarial noise from data. We consider a training dataset $\mathcal{X}_{\text{data}}$ that consists of data instances $\mathbf{x}_{\text{data}} \in \mathbb{R}^d$ where d denotes the dimension of the data space. Corresponding class labels (one-hot vectors) are denoted by $\mathbf{y}_{\text{data}} \in \mathbb{R}^c$ in a set of classes C where c is the number of classes.

We then consider a target classifier M_t which is the model that an attacker wants to deceive. We introduce a source classifier M_s which learns the decision boundaries on the

TABLE 1. Neural network architectures used for classifiers.

A	B	C	D	E
Conv(64, 5×5, 1)	Conv(128, 3×3, 1)	Conv(64, 5×5, 1)	Conv(128, 3×3, 1)	Dropout(0.2)
ReLU	ReLU	ReLU	ReLU	Conv(64, 8×8, 2)
Conv(64, 5×5, 2)	Conv(64, 3×3, 2)	Conv(128, 5×5, 1)	Conv(256, 3×3, 1)	ReLU
ReLU	ReLU	ReLU	ReLU	Conv(128, 6×6, 2)
Dropout(0.25)	Dropout(0.25)	Conv(256, 5×5, 2)	Conv(512, 3×3, 2)	ReLU
FC(128)	FC(128)	ReLU	ReLU	Conv(128, 5×5, 1)
ReLU	ReLU	Dropout(0.25)	Dropout(0.25)	ReLU
Dropout(0.5)	Dropout(0.5)	FC(512)	FC(512)	Dropout(0.5)
FC(10) + Softmax	FC(10) + Softmax	ReLU	ReLU	FC(10) + Softmax
		Dropout(0.5)	Dropout(0.5)	
		FC(256)	FC(10) + Softmax	
		ReLU		
		FC(10) + Softmax		

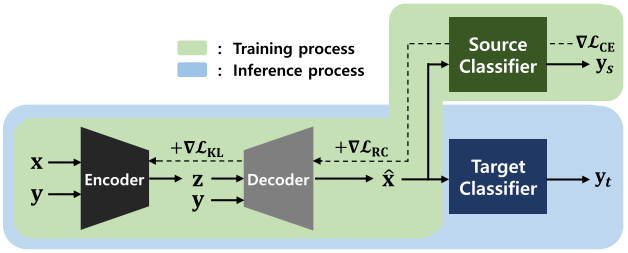


FIGURE 6. Overview of the PuVAE mechanism; The green region represents the training process, and the blue region denotes the inference process of PuVAE. The dotted line is the gradient flow in the training process. The parameters of the source classifier are not updated.

training data space. Here, M_s is independent of M_t. We also assume a set $\mathcal{X}_{\text{adv}}$ that consists of adversarial examples $\mathbf{x}_{\text{adv}} \in \mathbb{R}^d$. We define a set $\mathcal{X}$ that contains clean samples and adversarial examples. Instances $\mathbf{x}$ from the set $\mathcal{X}$ are predicted at inference time. The overview of the proposed method is shown in Figure 6.

A. TRAINING PROCESS OF PUVAE

PuVAE is comprised of an encoder network and a decoder network. The encoder receives a data-label pair and outputs the mean $\boldsymbol{\mu}$ and standard deviation $\boldsymbol{\sigma}$ of the Gaussian distribution on the latent space corresponding to the input label. This is expressed as follows:

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = \text{Encoder}(\mathbf{x}_{\text{data}}, \mathbf{y}_{\text{data}}). \tag{11}$$

Using $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ obtained from the encoder, the latent vector $\mathbf{z}$ on the latent space is sampled as follows:

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\epsilon} \cdot \boldsymbol{\sigma}, \tag{12}$$

$$\boldsymbol{\epsilon} \sim N(\mathbf{0}, \sigma_\epsilon \mathbf{I}), \tag{13}$$

where $\boldsymbol{\epsilon}$ denotes a random variable for the reparameterization trick, and σ_ϵ denotes a hyperparameter that controls the magnitude of the standard deviation used to sample the latent vector. In the experiments, we use $\sigma_\epsilon = 1$ in the training time as regular VAE.

In classification tasks, convolutional neural networks (CNNs) using pooling and strides are used to select useful features and to widen the receptive field. However, this selective nature of pooling and strides is a disadvantage for generative models, since the feature selection causes information loss. Therefore, we use a dilated CNN [34] as the encoder to get the latent vector $\mathbf{z}$. Dilated convolution inserts zeros in the filter, so that the receptive field is widened and information loss is effectively reduced.

The sampled $\mathbf{z}$ enters the decoder with the label and produces an output instance $\hat{\mathbf{x}}$ with the same dimension d as the input:

$$\hat{\mathbf{x}} = \text{Decoder}(\mathbf{z}, \mathbf{y}_{\text{data}}). \tag{14}$$

At the training time, PuVAE is trained to maximize the variational lower bound in a manner similar to cVAE. This process allows PuVAE to construct the mapping of legitimate data on the latent space. Loss functions from the encoder and decoder are calculated as follows:

$$\mathcal{L}_{\text{RC}} = -\mathbf{x}_{\text{data}} \log \hat{\mathbf{x}} - (1 - \mathbf{x}_{\text{data}}) \log(1 - \hat{\mathbf{x}}), \tag{15}$$

$$\mathcal{L}_{\text{KL}} = \boldsymbol{\mu}^2 + \boldsymbol{\sigma}^2 - \log(\boldsymbol{\sigma}^2) - 1, \tag{16}$$

where $\mathcal{L}_{\text{RC}}$ denotes the reconstruction loss function to minimize the difference between the input and output instances, and $\mathcal{L}_{\text{KL}}$ denotes the Kullback-Leibler (KL) divergence between the output latent vector distribution of the encoder and the normal distribution. The mean squared error (MSE) is one of the most widely used measures for reconstruction loss. However, we use the cross-entropy between $\mathbf{x}_{data}$ and $\hat{\mathbf{x}}$ as the reconstruction loss $\mathcal{L}_{\text{RC}}$. The derivatives of MSE and cross-entropy have similar forms when the sigmoid function is used in the last layer. In addition, the derivative of the sigmoid function in the derivative of MSE approaches 0 at both ends, which makes learning slow [35]. On the other hand, cross-entropy does not result in this problem.

Additionally, we use the cross-entropy calculated from a classifier as a loss function for PuVAE. Then, trained M_s is used to ensure that the output instance reflects the

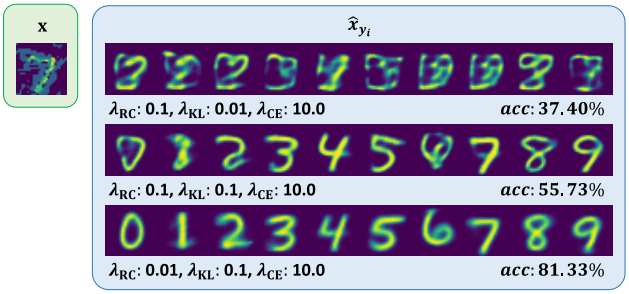


FIGURE 7. Comparison of coefficients for training PuVAE; an adversarial example is highlighted with the green box. The blue box represents the output samples when the input x is entered. Each row shows output images from PuVAE trained with different coefficient combinations. Each column shows images sequentially conditioned on class labels 0 – 9. *acc* denotes the classification accuracy of M_t using purified samples.

characteristic of the classes in C. The cross-entropy loss from M_s is as follows:

$$\mathbf{y}_s = M_s(\hat{\mathbf{x}}), \tag{17}$$

$$\mathcal{L}_{\text{CE}} = -\mathbf{y}_{\text{data}} \log \mathbf{y}_s - (1 - \mathbf{y}_{\text{data}}) \log(1 - \mathbf{y}_s). \tag{18}$$

Finally, PuVAE is trained using the following total loss:

$$\mathcal{L}_{\text{total}} = \lambda_{\text{RC}}\mathcal{L}_{\text{RC}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}} + \lambda_{\text{CE}}\mathcal{L}_{\text{CE}}, \tag{19}$$

where λ_{KL}, λ_{RC}, and λ_{CE} are coefficients for each loss functions. We conduct grid search with all combinations of λ_{RC}, λ_{KL}, and λ_{CE} in $\{0.01, 0.1, 1, 10, 100, 1000\}$, and obtain the best performance with the combination of 0.01, 0.1, and 10, respectively. Therefore, we set the coefficients to this combination.

B. GENERATING PURIFIED SAMPLES

At the inference time, PuVAE projects an input sample to the data manifolds of all classes in C as follows:

$$\hat{\mathbf{x}}_{\mathbf{y}_i} = \text{PuVAE}(\mathbf{x}, \mathbf{y}_i), \tag{20}$$

where $\mathbf{y}_i$ denotes the i-th class label in C to guide the input to the corresponding latent space, and $\hat{\mathbf{x}}_{\mathbf{y}_i}$ denotes a candidate for the purified sample. The inference follows the Equations (11), (12), and (13) as when training. σ_ϵ is used to sample the latent vector $\mathbf{z}$, and we perform a hyperparameter search on σ_ϵ among $\{0, 0.01, 0.1, 1, 10, 100\}$ for the inference. We find 0.1 as the most optimal value for σ_ϵ.

Then, the class label corresponding to the closest projection, $\mathbf{y}^*$, is selected as follows:

$$\mathbf{y}^* = \text{argmin}_{\mathbf{y}_i \in C} \; D(\mathbf{x}, \hat{\mathbf{x}}_{\mathbf{y}_i}), \tag{21}$$

where D denotes the distance measure to determine the closest projection. We use the root mean square error (RMSE) as the distance measure. The candidate generated with label $\mathbf{y}^*$ is the purified sample that goes into M_t as follows:

$$\mathbf{x}_{\text{purified}} = \hat{\mathbf{x}}_{\mathbf{y}^*}. \tag{22}$$

Finally, the purified sample is fed into the target classifier M_t. $\mathbf{y}_t$ is the predicted label of M_t as follows:

$$\mathbf{y}_t = M_t(\mathbf{x}_{\text{purified}}). \tag{23}$$

TABLE 2. Neural network architectures for PuVAE.

Encoder	Decoder
Dilated Conv(32, 7×7, 2)	FC(512)
ReLU	ReLU
Dilated Conv(32, 7×7, 2)	Deconv(32, 7×7, 2)
ReLU	ReLU
Dilated Conv(32, 7×7, 2)	Deconv(32, 7×7, 2)
ReLU	ReLU
FC(1024)	Deconv(32, 7×7, 2)
ReLU	Sigmoid
FC(1024)	
ReLU	
FC(64)	
Softplus (for σ only)	

The complete process of generating the purified sample using PuVAE is illustrated in Figure 2.

V. EXPERIMENTS

In this section, we determine the optimal setting for PuVAE, and present the defense performances of PuVAE against adversarial attacks. We use Tensorflow (1.12.0) for the experiments. A GPU, an NVIDIA TITAN V (12 GB), and a CPU, an Intel Xeon E5-2690 v4 (2.6 GHz), are used. We use MNIST [36] which is a hand-written digit dataset, Fashion-MNIST [37] which is a database of fashion images, and CIFAR-10 [38] which is an established computer-vision dataset used for object recognition. Each dataset consists of 50,000 training instances and 10,000 test instances. We normalize the data between 0 and 1.

We use FGSM, iFGSM, and the CW attack for the experiments. FGSM and iFGSM are generated with an adversarial perturbation size ε of 0.3 for the MNIST and Fashion-MNIST datasets, and 0.06 for the CIFAR-10 dataset. We set the maximum bound of the adversarial perturbation ε of iFGSM to 0.3 for the MNIST and Fashion-MNIST datasets, and to 0.06 for the CIFAR-10 dataset, and the size of small perturbation α is set to 0.03 for all datasets. We set the number of iterations of the CW attack to 100 for all datasets. The performance of defense mechanisms is measured by the accuracy of the target classifier.

The architectures of the encoder and decoder of PuVAE are shown in Table 2. Dilated Conv(n, $k \times k$, r) denotes a dilated convolution layer with n feature maps, filter size $k \times k$, and dilation rate r. Deconv(n, $k \times k$, s) denotes a deconvolution layer [39] with n feature maps, filter size $k \times k$, and stride s. FC(m) denotes a fully connected layer with m units. ReLU denotes the rectified linear unit. We use the first half of the last layer of the encoder, 32 output units, as $\boldsymbol{\mu}$ and the second half is passed to the softplus function to infer $\boldsymbol{\sigma}$. We use the architectures of Defense-GAN and the reformer network of MagNet as suggested in [19] and [20], respectively. The architectures of the classifiers used in our experiments are presented in Table 1. Architectures A and B are used for the MNIST and Fashion-MNIST datasets respectively, and

TABLE 3. Defense performance (accuracy) on the MNIST dataset under the white-box setting (%). The numbers in parentheses next to attack methods indicate the value of the hyperparameter of the attack methods.

Classifier	Attacks	No Attack	No Defense	Adv. Tr.	MagNet	Defense-GAN	PuVAE
A	FGSM (0.3)	99.51	12.46	78.57	26.90	**85.29**	81.33
	iFGSM (0.3)	99.51	0.00	91.72	72.48	87.40	**92.33**
	CW (100)	99.51	0.43	18.80	18.80	90.04	**90.80**
B	FGSM (0.3)	99.29	28.85	**88.49**	71.70	86.03	88.25
	iFGSM (0.3)	99.29	0.04	**93.56**	86.29	88.55	92.25
	CW (100)	99.29	0.59	19.20	19.10	90.76	**92.92**

TABLE 4. Defense performance (accuracy) on the Fashion-MNIST dataset under the white-box setting (%). The numbers in parentheses next to attack methods indicate the value of the hyperparameter of the attack methods.

Classifier	Attacks	No Attack	No Defense	Adv. Tr.	MagNet	Defense-GAN	PuVAE
A	FGSM (0.3)	93.46	4.06	11.51	11.71	56.67	**59.18**
	iFGSM (0.3)	93.46	0.00	50.31	34.70	65.11	**72.51**
	CW (100)	93.46	5.01	18.00	15.50	68.60	**80.59**
B	FGSM (0.3)	93.54	2.75	8.00	15.30	**58.20**	52.46
	iFGSM (0.3)	93.54	0.00	46.97	41.84	66.06	**71.04**
	CW (100)	93.54	4.87	16.70	16.70	67.62	**79.42**

TABLE 5. Defense performance (accuracy) on the CIFAR-10 dataset under the white-box setting (%). The numbers in parentheses next to attack methods indicate the value of the hyperparameter of the attack methods.

Classifier	Attacks	No Attack	No Defense	Adv. Tr.	MagNet	Defense-GAN	PuVAE
C	FGSM (0.06)	82.13	3.82	19.07	18.58	29.74	**33.71**
	iFGSM (0.06)	82.13	0.32	24.13	27.00	33.69	**35.49**
	CW (100)	82.13	9.88	**56.48**	40.13	38.13	36.36
D	FGSM (0.06)	80.30	3.19	15.00	18.98	30.11	**33.20**
	iFGSM (0.06)	80.30	0.43	21.51	29.95	32.47	**34.48**
	CW (100)	80.30	9.92	13.64	15.71	28.10	**31.70**

architectures C and D are used for the CIFAR-10 dataset. Architecture E is used as a substitute model for black-box attacks.

A. EFFECT OF COEFFICIENTS ON TRAINING PUVAE

Figure 7 demonstrates the characteristics of the generated samples based on the combinations of three coefficients λ_{RC}, λ_{KL}, and λ_{CE}. The first row of Figure 7 shows the generated images when the relative ratio of λ_{KL} is smaller than that of λ_{RC}. In this case, the constraint of the posterior distribution of the encoder is relieved. Thus, the encoder easily maps the input samples to the low likelihood area of the latent space. In addition, the encoder focuses on reconstructing the original input rather than mapping the input to prior latent space, so that the decoder restores even adversarial perturbations in the input images. These characteristics cause the decoder to generate strange images.

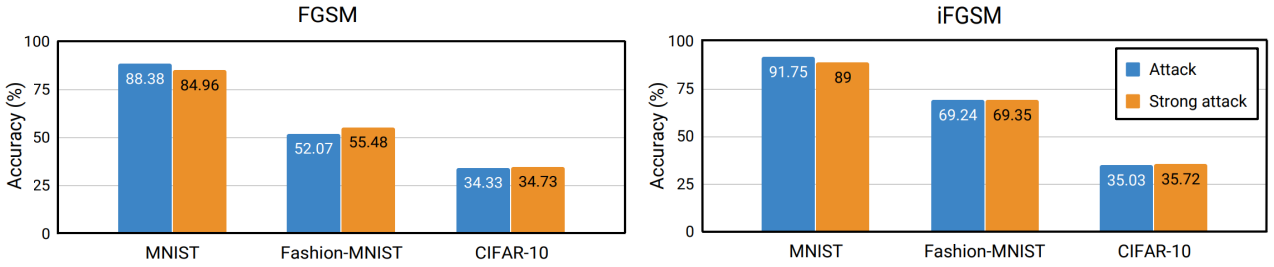
If λ_{KL} increases as in the second row of Figure 7, the generated images are organized by comparison with the first row, but is still affected by adversarial noise such as numbers 0, 1, 6 and 7. As a result of the greed search, PuVAE shows the best performance when the ratio between λ_{RC} and λ_{KL} is 1:10. The third row of Figure 7 is one of the configurations with the ratio. The performance results from the collaboration of $\mathcal{L}_{RC}$ and $\mathcal{L}_{KL}$. $\mathcal{L}_{KL}$ constraints the mapped area of input images for the decoder to generate images similar to the training data. $\mathcal{L}_{RC}$ makes the generated images maintain their morphological structure.

B. DEFENSE PERFORMANCE

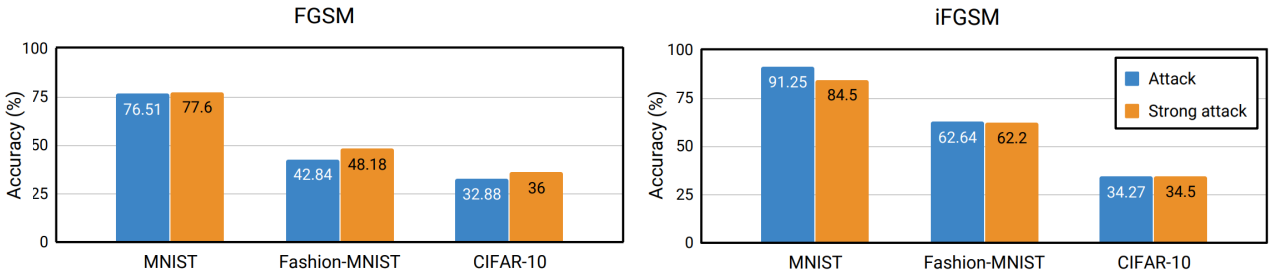
In this section, we compare the defense performance of PuVAE with the defense ability of adversarial training, MagNet and Defense-GAN. To obtain the best performance of Defense-GAN, we set the number of iterations to 200 and the number of candidates to 20. We alternately switch the architectures of the source and target classifiers. For example, when the architecture of the target classifier is A, architecture B is used as the source classifier, and vice versa.

TABLE 6. Defense performance (accuracy) on the Fashion-MNIST dataset under the black-box setting (%). The numbers in parentheses next to attack methods indicate the value of the hyperparameter of the attack methods.

Classifier	Attacks	No defense	MagNet	Defense-GAN	PuVAE
A	FGSM (0.3)	36.32	33.92	**66.94**	44.62
	iFGSM (0.3)	68.18	67.7	68.07	**77.76**
	CW (100)	21.49	**89.52**	69.39	81.32
B	FGSM (0.3)	34.13	32.51	**66.25**	45.19
	iFGSM (0.3)	62.76	60.84	67.64	**74.91**
	CW (100)	16.1	76.95	68.36	**81.22**



(a) Architecture A is used for the source model, and architecture B is used for the target model.



(b) Architecture B is used for the source model, and architecture A is used for the target model.

FIGURE 8. Comparison of defense performance between white-box and strong white-box attacks on the MNIST, Fashion-MNIST, and CIFAR-10 datasets.

Tables 3, 4, and 5 show the performances of defense methods against white-box attacks on the MNIST, Fashion-MNIST, and CIFAR-10 datasets respectively.

As shown in Table 3, the performance of PuVAE exceeds that of MagNet on all the attacks and is comparable to that of Defense-GAN. Adversarial training is also comparable with our method in FGSM and iFGSM despite of a very low performance in the CW attack. Since we use the gradients from M_t for adversarial training, it is robust against FGSM based attacks (FGSM and iFGSM), but weak against the other attack (CW). As shown in Table 4, PuVAE shows the best performance against iFGSM and the CW attack in both architectures. Even though iFGSM is a strong attack when there is no defense, the proposed model effectively defends the attack.

In general, the defense performances on the CIFAR-10 dataset shows overall low accuracy as shown in Table 5. Although adversarial training exhibits the best performance at a certain setting, it shows unstable results depending on models and attacks. However, PuVAE shows the best performance in various attacks and model architectures. Our method also exhibits a robust performance in settings where it is not first, indicating that it possesses a general defense ability across various attacks.

We compare the defense performance of PuVAE and purifier models (MagNet and Defense-GAN) against black-box attacks in the Fashion-MNIST dataset. As shown in Table 6, Defense-GAN shows better performance for FGSM than PuVAE, but PuVAE outperforms Defense-GAN for the iFGSM and CW attacks.

C. DEFENSE PERFORMANCE AGAINST STRONG WHITE-BOX ATTACKS

To verify the robustness of PuVAE, we assume strong white-box attacks where the attacker is aware of the architecture, parameters, and hyperparameters of PuVAE as well as the target network. The MNIST, Fashion-MNIST and CIFAR-10 datasets are used in this experiment. We use architecture A and B as the source and target classifier, respectively. Figure 8 shows the performances against white-box FGSMs and strong white-box FGSMs. The blue bar represents a defense performance against a white-box attack,

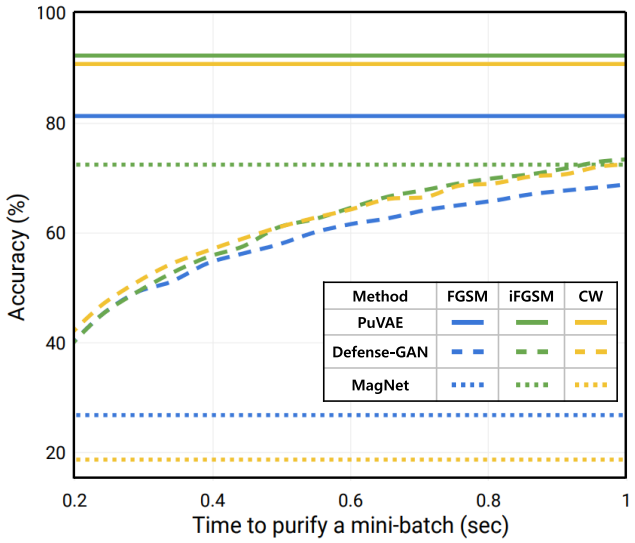


FIGURE 9. Performance (accuracy) comparison of defense methods in a reasonable time limit on a mini-batch of 128 MNIST images.

and the orange bar represents a defense performance against a strong white-box attack. Experimental results show that the classification performance is slightly decreased, or even increased when adversary has additional information on PuVAE. This suggests that PuVAE successfully projects adversarial examples onto the manifold of true data, so that there is little benefit for the adversary to acquire additional information of PuVAE.

D. PERFORMANCE IN A REASONABLE TIME CONSTRAINT

Defense mechanisms including Defense-GAN, MagNet, and PuVAE purify adversarial examples in a pre-processing manner. In contrast to PuVAE and MagNet, Defense-GAN takes a significant amount of time to derive the maximum performance. While Defense-GAN takes approximately 14.8 seconds, PuVAE takes 0.114 seconds to purify a mini-batch with 128 MNIST images, allowing nearly 130 times faster inference as shown in Table 7. Although, MagNet takes only 0.01 seconds to purify a mini-batch, it is still vulnerable to adversarial attacks.

As an example of real-world applications, autonomous driving needs rapid and accurate object detection. However, the object detection model is vulnerable to adversarial attacks that can cause a severe accident. Therefore, a defense method that works in tandem with the object detection model is essential. For applicability of the defense method, the defense performance needs to be measured on a reasonable time limit. In the experiments, we set the time limit as one second.

Figure 9 represents performance comparison of defense methods in time limit. The solid lines show the performances of PuVAE, the dotted lines show the performances of MagNet and the dashed lines show the performances of Defense-GAN. Each color denotes a different attack method. We use a mini-batch with 128 MNIST images. PuVAE performs

TABLE 7. Inference time comparison of defense methods on a mini-batch of 128 MNIST images.

Method	MagNet	Defense-GAN	PuVAE
Time (seconds)	0.01	14.80	0.11
Ratio	0.09	134.55	1

TABLE 8. Accuracy of defense methods on the MNIST dataset within one second (%).

Method	FGSM	iFGSM	CW
Adv. Tr.	78.57	91.72	18.80
MagNet	26.90	72.48	18.80
Defense-GAN	69.25	73.85	72.79
PuVAE	**81.33**	**92.33**	**90.80**

with one inference, and thus the performance of PuVAE is superior to that of Defense-GAN within the specified time limit. Since Defense-GAN creates a hidden vector iteratively using the gradient-based optimization process, the performance increases with time. However, it is unable to reach the performance of PuVAE in the given time limit. Therefore, PuVAE is more efficient than the state-of-the-art method for real-time applications. MagNet also performs with one inference, but its accuracy is lower than that of PuVAE.

It is unfair to compare PuVAE and Defense-GAN without time constraint because the inference time of Defense-GAN significantly exceeds that of PuVAE. As shown in Table 8, the performance of Defense-GAN is significantly lower than its maximum performance in reasonable time limit. Therefore, PuVAE is more practical in real-world scenarios because it exhibits the highest performance within a reasonable time condition.

VI. CONCLUSION

In this paper, we propose PuVAE, a novel VAE-based defense method that effectively purifies adversarial attacks. PuVAE is robust against various attacks and overcomes the disadvantages of adversarial training. The performance of PuVAE is also comparable to the best performance of Defense-GAN. In addition, PuVAE significantly outperforms Defense-GAN given a reasonable time limit. We demonstrate the advantages of the proposed method on various datasets and adversarial attacks. For future work, we plan to apply our method to real-time applications such as autonomous-driving, face identification, and surveillance systems.

ACKNOWLEDGMENT

(Uiwon Hwang, Jaewoo Park, and Hyemi Jang contributed equally to this work.)

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[2] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Sep. 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[3] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.
[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[5] U. Hwang, S. Choi, H.-B. Lee, and S. Yoon, "Adversarial training for disease prediction from electronic health records with missing data," Nov. 2017, *arXiv:1711.04126*. [Online]. Available: https://arxiv.org/abs/1711.04126
[6] J. Yoo, Y. Hong, Y. Noh, and S. Yoon, "Domain adaptation using adversarial learning for autonomous navigation," Dec. 2017, *arXiv:1712.03742*. [Online]. Available: https://arxiv.org/abs/1712.03742
[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Dec. 2013, *arXiv:1312.6199*. [Online]. Available: https://arxiv.org/abs/1312.6199
[8] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–11.
[9] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
[10] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," vol. 1, no. 2, p. 3, Feb. 2016, *arXiv:1602.02697*. [Online]. Available: https://arxiv.org/abs/1602.02697
[11] H. Bae, J. Jang, D. Jung, H. Jang, H. Ha, and S. Yoon, "Security and privacy issues in deep learning," Jul. 2018, *arXiv:1807.11655*. [Online]. Available: https://arxiv.org/abs/1807.11655
[12] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," May 2016, *arXiv:1605.07277*. [Online]. Available: https://arxiv.org/abs/1605.07277
[13] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," May 2017, *arXiv:1705.07204*. [Online]. Available: https://arxiv.org/abs/1705.07204
[14] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," Jun. 2017, *arXiv:1706.06083*. [Online]. Available: https://arxiv.org/abs/1706.06083
[15] J. Buckman, A. Roy, C. Raffel, and I. Goodfellow, "Thermometer encoding: One hot way to resist adversarial examples," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–22.
[16] C. Guo, M. Rana, M. Cisse, and L. van der Maaten, "Countering adversarial images using input transformations," Oct. 2017, *arXiv:1711.00117*. [Online]. Available: https://arxiv.org/abs/1711.00117
[17] G. S. Dhillon, K. Azizzadenesheli, Z. C. Lipton, J. Bernstein, J. Kossaifi, A. Khanna, and A. Anandkumar, "Stochastic activation pruning for robust adversarial defense," Mar. 2018, *arXiv:1803.01442*. [Online]. Available: https://arxiv.org/abs/1803.01442
[18] C. Xie, J. Wang, Z. Zhang, Z. Ren, and A. Yuille, "Mitigating adversarial effects through randomization," Nov. 2017, *arXiv:1711.01991*. [Online]. Available: https://arxiv.org/abs/1711.01991
[19] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–17.
[20] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 135–147.
[21] Y. Song, T. Kim, S. Nowozin, S. Ermon, and N. Kushman, "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples," Oct. 2017, *arXiv:1710.10766*. [Online]. Available: https://arxiv.org/abs/1710.10766
[22] K. Xu, S. Liu, P. Zhao, P.-Y. Chen, H. Zhang, Q. Fan, D. Erdogmus, Y. Wang, and X. Lin, "Structured adversarial attack: Towards general implementation and better interpretability," in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–21. [Online]. Available: https://openreview.net/forum?id=BkgzniCqY7
[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," Nov. 2017, *arXiv:1611.01236*. [Online]. Available: https://arxiv.org/abs/1611.01236
[24] T. Miyato, S.-I. Maeda, M. Koyama, K. Nakae, and S. Ishii, "Distributional smoothing with virtual adversarial training," Jul. 2015, *arXiv:1507.00677*. [Online]. Available: https://arxiv.org/abs/1507.00677
[25] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," Jul. 2016, *arXiv:1607.02533*. [Online]. Available: https://arxiv.org/abs/1607.02533
[26] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," Nov. 2015, *arXiv:1511.04599*. [Online]. Available: https://arxiv.org/abs/1511.04599
[27] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.
[28] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," Nov. 2015, *arXiv:1511.04508*. [Online]. Available: https://arxiv.org/abs/1511.04508
[29] N. Papernot *et al.*, "Technical report on the cleverhans v2.1.0 adversarial examples library," Oct. 2016, *arXiv:1610.00768*. [Online]. Available: https://arxiv.org/abs/1610.00768
[30] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 8, pp. 1798–1828, Aug. 2013.
[31] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," Dec. 2013, *arXiv:1312.6114*. [Online]. Available: https://arxiv.org/abs/1312.6114
[32] K. Sohn, X. Yan, and H. Lee, "Learning structured output representation using deep conditional generative models," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2015, pp. 3483–3491.
[33] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2223–2232.
[34] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," Nov. 2015, *arXiv:1511.07122*. [Online]. Available: https://arxiv.org/abs/1511.07122
[35] M. A. Nielsen, *Neural Networks and Deep Learning*, vol. 25. San Francisco, CA, USA: Determination Press, 2015.
[36] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[37] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," Aug. 2017, *arXiv:1708.07747*. [Online]. Available: https://arxiv.org/abs/1708.07747
[38] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[39] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," May 2015, *arXiv:1505.04366*. [Online]. Available: https://arxiv.org/abs/1505.04366

UIWON HWANG received the B.S. degree in biomedical engineering from Korea University, Seoul, South Korea, in 2016. He is currently pursuing the Ph.D. degree in electrical and computer engineering with Seoul National University, Seoul. His research interests include deep generative models, biomedical data science, and machine learning.

JAEWOO PARK received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2017, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include image processing, computer vision, and machine learning.

HYEMI JANG received the B.S. degree in electrical and computer engineering from Seoul National University, Seoul, South Korea, in 2017, where she is currently pursuing the Ph.D. degree in electrical and computer engineering. Her research interests include deep learning, security of neural networks, and continual learning.

SUNGROH YOON (S'99–M'06–SM'11) received the B.S. degree in electrical engineering from Seoul National University, South Korea, in 1996, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, CA, USA, in 2002 and 2006, respectively. From 2016 to 2017, he was a Visiting Scholar with the Department of Neurology and Neurological Sciences, Stanford University. He held research positions at Stanford University and Synopsys, Inc., Mountain View, CA, USA. From 2006 to 2007, he was with Intel Corporation, Santa Clara, CA, USA. He was an Assistant Professor with the School of Electrical Engineering, Korea University, from 2007 to 2012. He is currently a Professor with the Department of Electrical and Computer Engineering, Seoul National University. His current research interests include machine learning and artificial intelligence. He was a recipient of the SNU Education Award, in 2018, the IBM Faculty Award, in 2018, the Korean Government Researcher of the Month Award, in 2018, the BRIC Best Research of the Year, in 2018, the IMIA Best Paper Award, in 2017, the Microsoft Collaborative Research Grant, in 2017, the SBS Foundation Award, in 2016, the IEEE Young IT Engineer Award, in 2013, and many other prestigious awards.

NAM IK CHO received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, South Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate with the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was an Assistant Professor of electrical engineering with University of Seoul. In 1999, he joined the Department of Electrical and Computer Engineering, Seoul National University, where he is currently a Professor. His research interests include image processing, adaptive filtering, digital filter design, and computer vision.