# MAD-VAE: MANIFOLD AWARENESS DEFENSE VARIATIONAL AUTOENCODER

**Dingsu Wang**
New York University Shanghai
dw1920@nyu.edu

**Frederick Morlock**
New York University Shanghai
fm1391@nyu.edu

**ABSTRACT**

Although deep generative models such as Defense-GAN [1] and Defense-VAE [2] have made significant progress in terms of adversarial defenses of image classification neural networks, methods have been found to circumvent these defenses [3][4]. Based on Defense-VAE, in our research we introduce several methods to improve the robustness of defense models. The methods introduced in this paper are very straight forward yet very powerful. With extensive experiments on MNIST [5] data set, we have demonstrated the effectiveness of our algorithms against different attacks.

***K*eywords** Adversarial Defenses, Generative Models

## 1 Introduction

With the development of deep learning in recent years, research about image classification have made significant breakthroughs. Models such as *ResNet* [6] and *VGG* [7] as well as their variants have achieved amazing accuracy on image classification tasks. Despite their high accuracy on image classification, neural network based models are often vulnerable to misclassification given a well-crafted input by an adversary. The differences between these *adversarial examples* and original data are often imperceptible to humans, but result in misclassification by the model [8] [9].

Recent research has considered two main attack methods: White-Box attack and Black-Box attack. Under the White-Box attack scenario, the attackers have access to the classification model as well as the pretrained parameters. Under the Black-Box attack situation, the attackers don't have any information about the classification model [10].

Along with attacks, various defenses mechanisms have also been proposed in recent years. The defenses methods can mainly be divided into three groups [10]: (1) obfuscating the gradient of the classification model [2][1] [8][11][12], (2) robust optimization such as such as training the classification model with adversarial examples [8] and (3) adversary detection before feeding images to the classification model [13][14].

In our research, we focus on the first kind of defense mechanism and improve upon the Defense-VAE [2] model. The idea of Defense-VAE is to use VAE's ability to learn the distribution of input data. After learning the distribution of data, Defense-VAE transfers the data back to the data manifold. In the original VAE framework [15], the model consists of two parts: an encoder $E$ which maps the input data onto a latent distribution, and a decoder $D$ which tries to reconstruct the input data based on the variable $z$ sampled from the latent distribution. Defense-VAE leverages this idea in their defense model and proves that it is a robust way to prevent against attacks. Unfortunately, although generative defense models such as Defense-GAN [1] and Defense-VAE [2] have achieved decent results on prevent certain attacks, they have been shown to still be venerable to carefully crafted adversarial examples [3].

The following paper consists of following parts: for theoretical background and related works, please see Appendix A; in Section 2, we will introduce the research work we are focusing on; Section 3 will be about the experiments and results while Section 4 discusses future works and concludes the paper. There are additional plots and tables in the remainder of the Appendix that were not included in the main body of the paper.

## 2 Proposed MAD-VAE

We propose several training methods and additional loss functions, which may help Defense-VAE to be more robust against attacks such as the Overpowered Attack. We call our model Manifold Awareness Defense-VAE because our

methods are trying to capture the topological structure data manifold as well as the latent variables. In the following subsections, we will first talk about the motivation of our research ideas and then describes our methods accordingly.

### 2.1 Motivation

In order to solve this problem of adversarial attack, we formulate our idea in the following way: we want the encoded $z$ to represent the data well while having the decoded $G(z)$ be correctly classified. Since $z_i$ follows a certain distribution for each class $i$, we want the points $z_i^j$ sampled from this distribution will be close to each other while the cluster of these points are well separated from the cluster with other class label. Ideally, our model should be able to encode the adversarial data $X'$, where the benign $X$ data has label $i$, on to or close to the distribution $z_i$. Therefore, when we sample from the distribution or trying to find $z_i^j$ that can generate similar data, it will be less likely to generate data that lie in the distribution cluster of other labels and we can make sure the output of our model can be correctly classified.

### 2.2 Algorithm

Aiming to achieve the goal we propose above, we introduce several loss functions to make our Defense-VAE model more robust and aware of the underlying topology of the data manifold.

#### 2.2.1 Classification Loss

When the classifier gets an input data $X$, it will extract certainly features from $X$ and make classification based on them. Since our goal is to make sure that the output of our model can be correctly classified, one straight forward way to improve the model is to having a classification loss on the output of the VAE $G(z)$. This idea is from GAN (Generative Adversarial Networks) [16] where we have a generative model $G$ to generate data and a discriminator $D$ to make sure the generated data is what we want.

Specifically in our case, we have the VAE as our generative model and the pretrained classifier to be our discriminator model. During training, we fix the parameters of the classifier and feed the output of VAE $G(z)$ to it and get a classification loss $L_c$. The new loss function of our model is as follows:

$$L = L_r + \beta \cdot L_{kl} + \alpha \cdot L_c$$

where $L_r$ is the reconstruction loss and $L_{kl}$ is the KL-divergence loss in the *ELBO* equation. $\alpha$ and $\beta$ are two hyperparameters of the weight on different loss.

By doing so, we hope the classification loss can help to push the VAE model generate features that are sensitive to the classifier, and at the same time, force the latent $z$ to lie in the cluster of $z_i$ for each class $i$.

#### 2.2.2 Proximity and Distance Loss

According to Zhao et al. [17], due to the nature of MNIST, it would be hard for a model like our own to learn a clear separation between classes and as a result the latent distributions of $z$ will certainly mix with each other. In order to solve this problem, we decide to add another explicit loss to constrain the latent variables. Similar to the idea from the research conducted by Mustafa et al. [18], we try to learn the center $c_i$ of the cluster of $z_i$ for each class $i$ and, at the same time, keep the clusters away from each other. The proposed loss of function can be described as follows:

$$\begin{aligned} L_p &= ||z_i - c_i||_2 \\ L_d &= \frac{1}{k-1} \cdot \sum_{j \neq i} \left( ||z_i - c_j||_2 + ||c_i - c_j||_2 \right) \\ L_{pd} &= \sum_i \left[ \alpha \cdot L_p - \sigma \cdot L_d \right] \end{aligned}$$

where $k$ is total number of labels in the dataset and the distances are computed by the euclidean distances. $\alpha$ and $\sigma$ are the weights between the proximity loss and distance loss respectively. Therefore, the loss function for our VAE model becomes:

$$L = L_r + \beta \cdot L_{kl} + L_{pd}$$

where the symbols are the same as those above.

The idea of this is to force the latent variable $z_i$ with the same label $i$ to gather together towards the center $c_i$ while make sure $c_i$ and $z_i$ is farther away from the center of cluster of $z_j$ where $j \neq i$. This idea is straight forward and we borrow the illustration image from the paper [18] here:

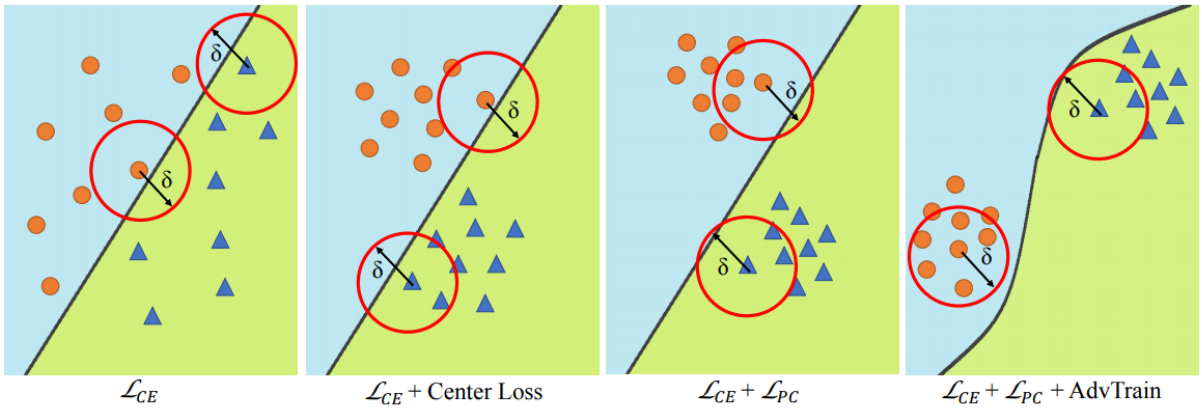



Figure 1: Comparison of different training methods. [18]

Although we are not applying adversarial training in our algorithms, the inputs to our model are always the adversarial examples with different attacks, performed using various parameter settings. Thus, we are pushing the decision boundaries between each cluster to better separate the classes as is shown in Figure 1.

#### 2.2.3 Combined Loss

Since we have discussed two new defense loss functions in the subsections above, it would be natural to think of a way to combine them together and utilize both their advantages. A naive approach to the aggregation of the two defense loss functions would be to design a weighted sum. To do so, we use the combined loss below for our VAE model:

$$L = L_r + \beta \cdot L_{kl} + \gamma \cdot L_c + L_{pd}$$

where $L_r$ is the reconstruction loss and $L_{kl}$ is the KL-divergence loss in the $ELBO$ equation. $\beta$ and $\gamma$ are two hyperparameters of the weight on different loss.

## 3 Experiments

We compare our methods with the original Defense-VAE model, however, it may be noted that our model does not use any of the Defense-VAE's code base. Surprisingly, we have been unable to reproduce the results detailed in the Defense-VAE paper. Despite the inability to reproduce their results, after careful analysis of both our code and theirs, we are confident that our base code is functionally equivalent to the code for Defense-VAE.

We evaluate our defensive model under the FGSM, Rand-FGSM and CW white-box attacks as well as three other attack methods not included in our training process: PGD [19], momentum iterative FGSM [12] and single pixel attack [20]. The adversarial examples of these attacks can be seen in the appendix. Our codes are based on PyTorch [21] and use the open source package AdverTorch [22] for performing attack methods. We use machines equipped with NVIDIA GeForce GTX 1080 Ti GPUs.

We use the MNIST dataset [5] for our experiments, which contains 60, 000 training images and 10, 000 testing images. We generate a big training dataset using three attack methods: FGSM, Rand-FGSM and CW with three different parameter settings for each one of them. We then use another parameter setting to generate the validation dataset to evaluate the performance of our models, and we use the testing data set to get our test dataset for all the six attack algorithms mentioned above. Since the attack generating process contains a lot of random computations in it, it is rare for non-empty intersections of different dataset to occur.

Due to time limitations and the computational complexity of the models, we have not been able to fully reproduce the experiments conducted in Defense-VAE [2] and Defense-GAN [1], however, we have based our defense model based on the architecture of Defense-VAE [2]. Our defensive model's structure can be found in Table 5 in the Appendix. In our experiments, we choose the classifier MagNet [23] to be our classification model. MagNet's model structure can be found in Table 6 in the Appendix.

The original Defense-VAE model converges after 5 epochs while our model, with the new loss functions, usually converges after 10 epochs. In order to achieve the best performance, we validated our model on a validation data set for

different parameter settings. From our experiments on the validation set, we chose the weight of our loss functions as: $\alpha = 0.01$, $\sigma = 0.00001$, $\beta = 0.1$ and $\gamma = 0.1$. The other parameters setting are kept the same as Defense-VAE paper. We use the Adam optimizer for our gradient descent with a exponential learning rate scheduler.

### 3.1 Results on White-Box Attack

In this section, we will present the results on White-Box attack using six different attack methods: FGSM, Rand-FGSM, CW, MI-FGSM, PGD and Single Pixel. All the attacked images are generated using the AdverTorch [22] package with default parameters. The results are shown in Table 1.

We note that the attacks are successfully fooling the classifier except for the Single Pixel attack, perhaps because of internal bugs within the AdverTorch packages. The table demonstrated that our methods make the original Defense-VAE robust under all the six attacks, and the model with proximity and distance loss function outperformed the others on most of the situations. We did not fine-tune the classifier based on the output of out model like what was done in the Defense-VAE paper. While we were unable to reproduce the Defense-VAE model in terms of accuracy, what we have been able to do is beat Defense-VAE in our testing using our topologically aware training.

| Attack | No Attack | No Defense | Vanilla | Classification | Proximity and Distance | Combined |
|---|---|---|---|---|---|---|
| **FGSM** | 0.9931 | 0.1316 | 0.7097 | 0.7107 | **0.7413** | 0.7325 |
| **Rand-FGSM** | 0.9931 | 0.1521 | 0.7689 | 0.7833 | **0.7953** | 0.7922 |
| **CW** | 0.9931 | 0.0075 | 0.9567 | **0.9621** | 0.9433 | 0.9480 |
| **MI-FGSM** | 0.9931 | 0.0074 | 0.6821 | 0.6427 | **0.7112** | 0.5879 |
| **PGD** | 0.9931 | 0.0073 | 0.7530 | 0.7485 | **0.7789** | 0.6856 |
| **Single Pixel** | 0.9931 | 0.9977 | 0.9759 | 0.9922 | 0.9789 | **0.9928** |

Table 1: Classification accuracy of different models based on the FGSM, Rand-FGSM, CW, Momentum Iterative FGSM, PGD and Single Pixel White-Box attack on the classifier with the default parameters. The models are trained on the data generated using the first three attack methods while the other three attacks are not included in the training dataset.

In table 2, we provide the confusion matrix of the classification accuracy of Defense-VAE with proximity and distance loss function. We use the entire dataset which combines three White-Box attack algorithms: FGSM, Rand-FGSM and CW. It is obvious to notice that, images of digit $9$ is easy to be mis-classified as digit $4$, however, not too many images with digit $4$ fall into the category of $9$. The confusion matrix of other models can be seen in the appendix.

| **pred** \ **true** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2456 | 0 | 30 | 3 | 2 | 9 | 38 | 2 | 23 | 2 |
| **1** | 8 | 3357 | 77 | 19 | 37 | 18 | 31 | 120 | 48 | 62 |
| **2** | 26 | 4 | 2508 | 29 | 11 | 4 | 10 | 32 | 54 | 11 |
| **3** | 72 | 18 | 204 | 2680 | 6 | 219 | 13 | 69 | 281 | 49 |
| **4** | 10 | 4 | 44 | 7 | 2609 | 9 | 105 | 47 | 63 | 1073 |
| **5** | 68 | 3 | 10 | 171 | 8 | 2227 | 111 | 4 | 193 | 45 |
| **6** | 92 | 6 | 16 | 4 | 20 | 40 | 2497 | 0 | 24 | 3 |
| **7** | 10 | 1 | 159 | 50 | 20 | 6 | 3 | 2624 | 17 | 93 |
| **8** | 98 | 9 | 38 | 41 | 39 | 42 | 66 | 17 | 2087 | 57 |
| **9** | 100 | 3 | 10 | 26 | 194 | 102 | 0 | 169 | 132 | 1632 |
| **Accuracy** | 0.84 | 0.99 | 0.81 | 0.88 | 0.89 | 0.83 | 0.87 | 0.85 | 0.71 | 0.54 |

Table 2: Confusion matrix of accuracy of Defense-VAE with proximity and distance loss on entire test datasets (combining all the attacks).

### 3.2 Results on Black-Box Attack

In this section, we present our experiments of our models compared with the vanilla Defense-VAE on the FGSM Black-Box attack. As what is mentioned above, attackers will have no access to the classifier information under the Black-Box attack scenario. They thus training a substitute model for generating attacks. The detail of substitute models can be found in the table 7.

In the table 3, we present our classification results different substitute classification model from the Defense-VAE paper [2]. For all the experiments, we set the FGSM parameter $\epsilon$ to be $0.3$ which is the default value given by the original paper. From the table, we can spot that the Black-Box FGSM attack can reduce the classification accuracy of different classifiers to only $5$ percent. Compared to randomly choosing an output class, all models prove to be better than random. On the other hand, for all substitute models at least one of our defensive models out-performs Defense-VAE. The accuracy of our proximity and distance loss is the highest in most of the black-box attacks, although the accuracy is far from that of the substitute model with no attacks.

| Substitute | No Attack | No Defense | Vanilla | Classification | Proximity and Distance | Combined |
|---|---|---|---|---|---|---|
| A | 0.9939 | 0.0487 | 0.4212 | 0.4338 | **0.5879** | 0.4453 |
| B | 0.9925 | 0.0144 | 0.2771 | 0.2539 | **0.4048** | 0.3285 |
| C | 0.9938 | 0.0540 | 0.543 | 0.5667 | **0.6815** | 0.6060 |
| D | 0.9812 | 0.0173 | 0.4106 | 0.2932 | 0.2725 | **0.4155** |
| E | 0.9807 | 0.0155 | 0.4322 | 0.3177 | **0.4498** | 0.4442 |

Table 3: Classification accuracy of different models based on the FGSM Black-Box attack on various substitute models with $\epsilon = 0.3$.

### 3.3 Clustering

In this section, we will show the clustering of different variables using the Uniform Manifold Approximation and Projection (UMAP) [24] dimension reduction techniques. On the one hand, clustering of output images can give us a direct sense of the classification result, where images with same label tend to be close to each other. On the other hand, since we are applying the idea of clustering on the latent variables, it is helpful to use clustering figures to help us evaluate the results. Since we have four models in total, in the following plots we will denote vanilla Defense-VAE as $A$, Defense-VAE with combined loss as $B$, Defense-VAE with proximity and distance loss as $C$, and Defense-VAE with classification loss as $D$.

Since the experiments in the previous section show the strong performance of our Defense-VAE model endowed with the proximity and distance loss functions, due to the limited space we decided to only show the clustering plot of that model in this section. Additional plots for other defensive models and adversarial attacks can be accessed in the appendix.

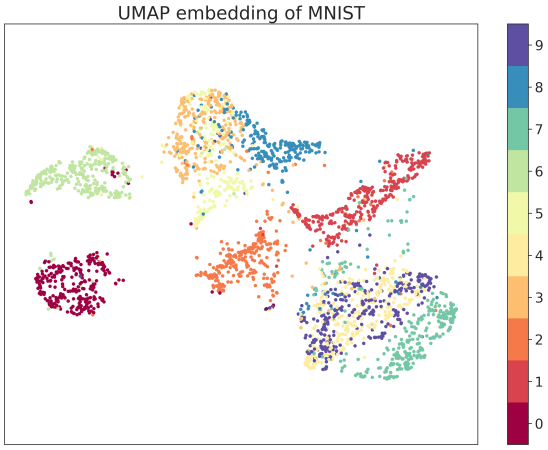
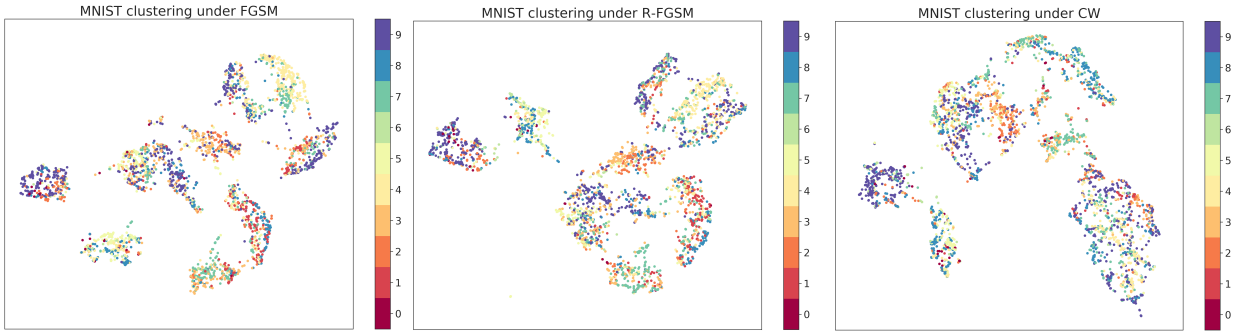



Figure 2: Clustering plots of the benign test dataset.

Figure 2 is the clustering of original MNIST test dataset, from which we are able to see that images of digit $6$ and $0$ are well separated from others although there have some points have similar characteristics and lie in the wrong clusters. The images of digit $9$ and $4$ are mixed with each other due to the similarities of certain features in the images, while other clusters are close to each other but still separated for a certain proportion.

Figure 3 is the MNIST clustering of data and predicted labels generated from three attacks: FGSM, Rand-FGSM and CW. From the figure we cannot spot any clear, homogeneous clustering which demonstrates the success of attacks on the data. If the attacks had failed, then the plot would show clustering of label colors.

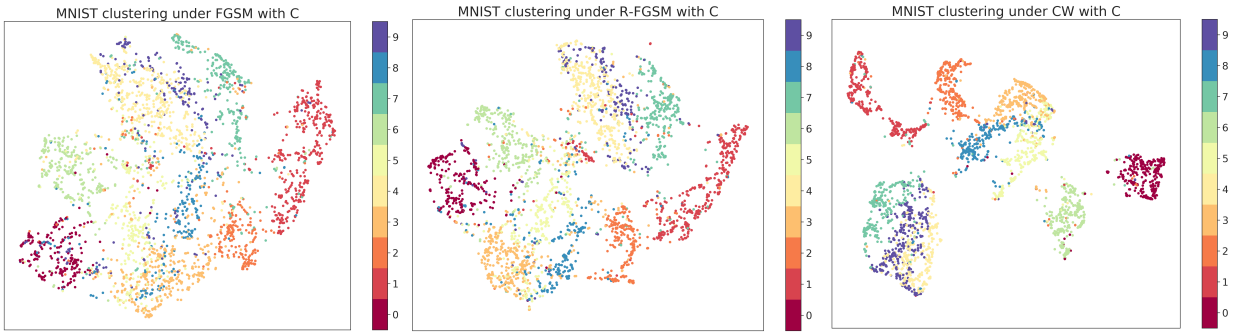



(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 3: Clustering plots for adversarial examples

In the following two groups of plots, we will show the clustering of the output of our Defense-VAE model with the proximity and distance loss function as well as the latent variables $z$. Since our goal is to achieve data manifold awareness during training, we are hoping to spot clustering in our data and latent variables.

Figure 4 shows the clustering result of the output from Defense-VAE with proximity and distance loss on adversarial data. The colored labels of Figure 4 are predicted using the classifier. From this figure, we are able to see that the output of our model is clearly clustered by color, whereas the adversarial input (Figure 3) is not. Although some of the data with different labels are still mixed, it demonstrate the potential ability of the model to remap data back to the original data manifold.




(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

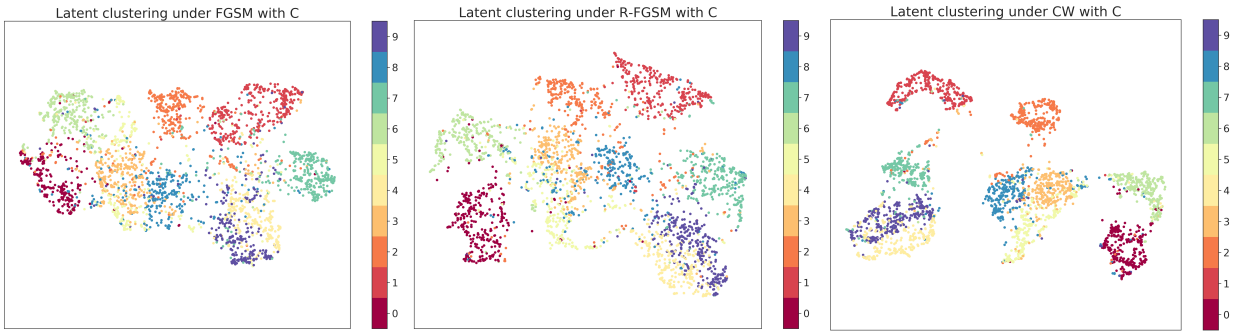
Figure 4: Clustering plots of output from model trained with proximity and distance loss across different attack methods.

Figure 5 shows the clustering of the latent variable $z$ in the proximity and distance defensive model. Since we are adding the proximity and distance loss with respect to the latent variables in our method, we would hope to see the clear separation of latent variables compared to the vanilla Defense-VAE. In fact, compared to the figures of latent variables of other models (including vanilla Defense-VAE), our model with the proximity and distance loss does not show significant improvement in clustering, although it does show subtly more distinct clustering.

### 3.4 Comments on Defense-VAE

Although all the experiments we have done are not based on the open source code provided by the author of Defense-VAE, we have carefully read through their codes and there are several comments that we want to make in this section.

The loss function they use is a little bit different from ours: they apply the reparameterization trick of calculating the KL-divergence loss while we directly use the KL-divergence function provided by PyTorch in the distributions module. Another interesting notice is that their default value of $\beta$ (weight for KL-divergence loss) in the code is 0, which makes their model to be a pure autoencoder instead of a variational autoencoder. Since we find it takes more than a week to run their codes successfully, we decided to use their model's file and loss function and train it on our own dataset. We get the results on the six FGSM White-Box attack in Table 4.




(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 5: Clustering plots for latent variables of the model trained with proximity distance loss given adversarial examples as input.

Compared to the result we have in Table 1, we can see that the accuracy score of the Defense-VAE trained with codes provided by the paper's authors performs worse on all the White-Box attacks compared to ours.

| Substitue | No Attack | No Defense | Defense-VAE |
|---|---|---|---|
| **FGSM** | 0.9931 | 0.1316 | 0.4358 |
| **Rand-FGSM** | 0.9931 | 0.1521 | 0.5337 |
| **CW** | 0.9931 | 0.0075 | 0.5024 |
| **MI-FGSM** | 0.9931 | 0.0074 | 0.1146 |
| **PGD** | 0.9931 | 0.0073 | 0.1726 |
| **Single Pixel** | 0.9931 | 0.9977 | 0.9977 |

Table 4: Classification accuracy of Defense-VAE with original code provided by the Defense-VAE's authors.

## 4 Conclusions and Future Works

In this paper, we propose the MAD-VAE with three different training loss functions to improve the robustness of Defense-VAE model. Our ideas do not limited to any model and can be applied to any generative model based defense algorithms. We empirically show that our methods are effective against various attack mechanisms.

There are definitely some improvements that could be made on the results of this paper. Due to time constraints, we are not able to run the Overpowered attack mentioned in the paper by Jalal et al. [3] which is a powerful attack algorithm aiming at the generative model based defense. Part of what the topological awareness written about by Jang et al. [25] is aimed to prevent such an attack, though we have not tested that theory. Therefore, it would greatly informative to see what the results of our methods are when encounter such strong attacks. Another aspect that worth delving into would be adapting our training methods to other models such as Defense-GAN. Since it is training a GAN is not an easy task, it was not feasible to complete it within our limited time-frame. Moreover, the choice of different hyperparameters is also worth tuning for a longer period of time. In order to achieve better performance, a larger training data set could be generated and could be used to better tune the hyperparameters.

## Acknowledgement

We are thankful for the help and insightful suggestions offered by Professor Shuyang Ling at New York University Shanghai.

## References


[1] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605*, 2018.

[2] Xiang Li and Shihao Ji. Defense-vae: A fast and accurate defense against adversarial attacks. *arXiv preprint arXiv:1812.06570*, 2018.

[3] Ajil Jalal, Andrew Ilyas, Constantinos Daskalakis, and Alexandros G. Dimakis. The robust manifold defense: Adversarial training using generative models, 2017.

[4] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples, 2018.

[5] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.

[6] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[7] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[8] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[9] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[10] Han Xu, Yao Ma, Haochen Liu, Debayan Deb, Hui Liu, Jiliang Tang, and Anil K. Jain. Adversarial attacks and defenses in images, graphs and text: A review, 2019.

[11] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2016.

[12] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum, 2017.

[13] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples, 2017.

[14] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *Proceedings 2018 Network and Distributed System Security Symposium*, 2018.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[17] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards deeper understanding of variational autoencoding models, 2017.

[18] Aamir Mustafa, Salman Khan, Munawar Hayat, Roland Goecke, Jianbing Shen, and Ling Shao. Adversarial defense by restricting the hidden space of deep neural networks. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[19] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[20] Nina Narodytska and Shiva Prasad Kasiviswanathan. Simple black-box adversarial perturbations for deep networks, 2016.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[22] Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. AdverTorch v0.1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019.

[23] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples, 2017.

[24] L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

[25] Uyeong Jang, Susmit Jha, and Somesh Jha. On need for topology-aware generative models for manifold-based defenses, 2019.

[26] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses, 2017.

[27] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[28] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning, 2016.

[29] Chuan Guo, Jacob R. Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Q. Weinberger. Simple black-box adversarial attacks, 2019.

[30] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017.

# Appendices

## A Related Work and Background Information

We have based our research on Defense-VAE and made some improvements using different loss functions and training methods. Before talking in detail about our work, we will discuss different attack and defense algorithms first as well as introduce VAE and Defense-VAE in detail. Finally, we will mention the problems and drawbacks of current research.

### A.1 Attack Methods

There are various attack algorithms haven been proposed. All of these attacks can be summarized in the following form: $\tilde{x} = x + \eta$ where $x$ is the original data and $\eta$ is the perturbations being added. Most of these attack methods aim to find the minimum perturbation $\eta$ which is undetectable by human beings but result in misclassification of the classifier.

#### A.1.1 White-Box Attacks

In the White-Box Attack scenario, the attackers will have full access to the classification model and the pretrained parameters. Therefore, the attackers can utilize the gradient of the model $J(\theta)$ to design attack algorithsm. Similar to the attack methods being used in Defense-GAN [1] and Defense-VAE [2], we are using three major attacks: FGSM (fast gradient sign method) [8], r-FGSM (random fast gradient sign method) [26] and CW (Carlini-Wagner attack) [27].

**Fast Gradient Sign Method** is proposed by Ian Goodfellow [8], where the attack is achieved by taking a single step in the direction of gradient of the classification model. The perturbation added to the image is $\eta = \epsilon sign(\nabla_x(\theta, x, y))$.

**Randomized Fast Gradient Sign Method** is an enhanced attack algorithm [26] based on the FGSM attack. Instead of directly getting the gradient on the original image data $x$, r-FGSM added random noise on the image $x' = x + \alpha \cdot sign(N(0^d, I^d))$, and then doing the FGSM attack on the generated image $x'$: $x^{adv} = x' + (\epsilon - \alpha) \cdot sign(J_{x'}(\theta, x', y_{true}))$.

**Carlini-Wagner attack** is a powerful attack [27] and it result in $0\%$ accuracy of the classifier under most of the circumstances. The attack is performed by solving the following optimization problems [27] [1]:

$$\begin{aligned} \min_{\delta \in R^n} &= ||\delta||_p + c \cdot f(x + \delta) \\ s.t. \quad & x + \delta \in [0, 1]^n \end{aligned}$$

where $f$ is an objective function that will result in the misclassification of the model and $c$ is a constant.

#### A.1.2 Black-Box Attacks

Under the scenario of Black-Box attack, the attackers will have no access to the classification model. Therefore, it is often much more difficult for the attackers to perform successful and efficient attack on the model. In most of recent research works, two main methods have been proposed [10]: (1) find substitution models which will resemble the classifier, then use the the original White-Box attack methods[28], and (2) another method is to use efficient query methods by estimating the gradient from model output or repeatedly randomly adding certain perturbations [29].

### A.2 Defenses Algorithms

In addition to the release of numerous attack methods there has also been extensive research into defense mechanisms. Although there are many of them, we will only introduce two major algorithms in the following subsections.

#### A.2.1 Adversarial Training

One of the major ways to make classification model more robust against different adversarial attacks is to augment the training data set. The process of training classification model on a integrated data set of benign data and adversarial examples generated using different adversarial models is called adversarial training [8]. Although this method has been proved to be somewhat useful for defending attacks from attacks that are used to augment data, it is still vulnerable when facing a different attack that is not included in the combined training data set.

### A.2.2 Deep Generative Models and Denoising

Another popular method is through leveraging the phenomenon of exploding and vanishing gradients in deep neural networks [10]. By first inputting the image into a generative model such as PixelDefend [30] or Defense-GAN [1] before performing classification, the combined model is deeper and thus is subject to exploding and vanishing gradients. Additionally, the defense model is able to transfer the adversarial example back to the benign data manifold and thus, these generative defensive models can be viewed as a certain type of purifier. This is the method that is utilized in Defense-VAE [2].

## A.3 Variational Autoencoders (VAE)

Variational Autoencoders (VAE) are powerful generative models based on the ideas of Autoencoders and variational inference. The model consists of an encoder $Q(z|X)$ and a decoder $P(X|z)$. The encoder tries to encode the information of input data $X$ onto a latent variable $z$, while the decoder aims to completely reconstruct the data $X$ using the latent variable $z$.

In order to maker sure our encoder successfully approximate the posterior $P(z|X)$, we should calculate the Kull-back–Leibler divergence (KL divergence or denoted by $KL$) between this two distributions.

$$KL\left[Q(z|X)||P(z|X)\right] = E\left[\log(Q(z|X)) - \log(P(z|X))\right]$$

Using Bayes' rule, we are able to get the objective function of VAE as follows:

$$\log(P(X)) - KL\left[Q(z|X)||P(z|X)\right] = E\left[\log(P(X|z))\right] - KL\left[Q(z|X)||P(z)\right]$$

We want to maximize the term on the left, since the KL divergence should be close to 0 if we want our $z$ to be good enough to reproduce the data. The two terms on the right hand side ($ELBO$: evidence lower bound) can be optimized using gradient descent: the former one measures how well we reconstruct the original data, and the latter one bounds the latent $z$'s distribution to the real probability distribution $P(z)$ which we usually assume it to be standard normal distribution.

## A.4 Defense-VAE

The original VAE model is not suitable to be used in adversarial defenses, since we do not want to reconstruct images still containing the information of perturbations. In order to solve this problem, Defense-VAE modifies the encoder and decoder as follows [2]:

$$z \sim Enc(\tilde{X}) = Q(z|\tilde{X}), \qquad X \sim Dec(z) = P(X|z)$$

where $\tilde{X} = X + \eta$ is the adversarial example created by adding perturbation $\eta$ on the original data $X$. Since the distribution of the input data has been changed, the objective function of Defense-VAE changes as follows:

$$ELBO = E_{Q(z|\tilde{X})}\left[\log(P(X|z))] - KL[Q(z|\tilde{X})||P(z)\right]$$

where the input of Defense-VAE are the adversarial examples while the output are the corresponding benign data.

The training process of Defense-VAE is very straight forward: we just need to use different attack algorithms to generate adversarial example based on the clean data thus creating a training pair. In reality, since we can use any attack methods with various parameters, we are able to generate large number of training data. The training pipeline for Defense-VAE is shown in Figure 6.

## A.5 Problems and Drawbacks of Current Research

Despite the best efforts of the respective authors, the state-of-art gradient obfuscating defense algorithms such as Defense-GAN [1] and Defense-VAE [2] are still vulnerable to adversarial attacks [3] [4]. Given a certain input image $X$, an attack can find another input $X'$ which is $\epsilon$-close to the original data $X$ that the classification output $C(X)$ and $C(X')$ are significantly different. Therefore, if such $(X, X')$ pair exists, then we can find a pair of $z$ accordingly: $(z, z')$ where $X \sim G(z)$ and $X' \sim G(z')$, yet $C(G(z))$ and $C(G(z'))$ are much different. Such attack method can be formulated as the following optimization problem:

$$\sup_{z,z'} L(C(G(z)), C(G(z'))),$$
$$s.t. ||G(z') - G(z)||_2^2 \leq (2\eta + \epsilon)^2$$

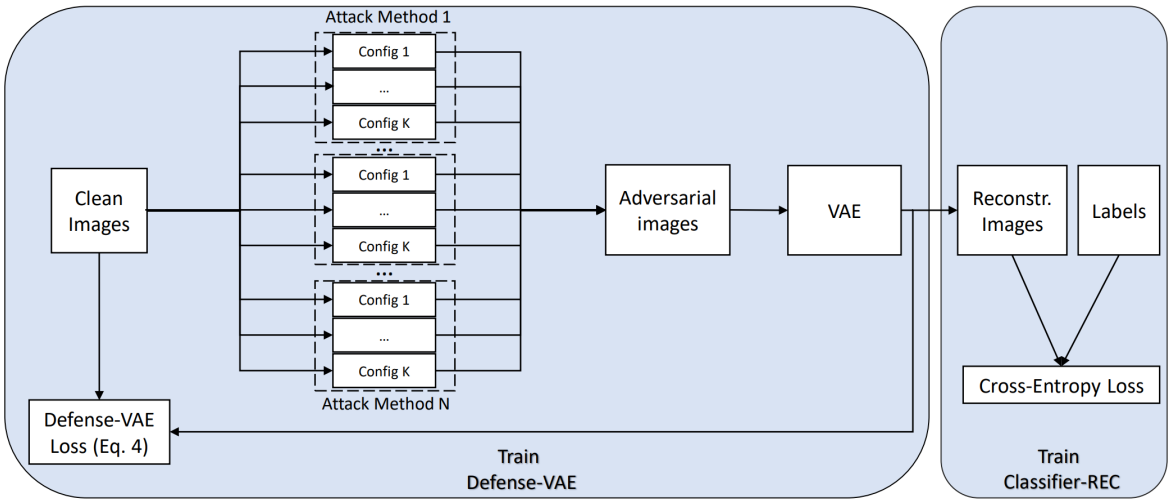



Figure 6: Training pipeline of Defense-VAE. [2]

where $L$ is some loss function, $\epsilon$ is the perturbation distance bound and $\eta$ is the distance bound between the generated data manifold $G$ and the real image manifold.

In their paper, Jalal et al. [3] proposed the Overpowered Attack which is based on this formulation, resulting in decreasing the accuracy of Defense-GAN to only $3\%$. In addition, they also proposed the according defense methods: adversarial training with the Overpowered Attack, which makes the model powerful against PGD (projected gradient descent attack, another popular White-Box attack) [19].

### A.6 Topological Structures in Adversarial Defense

In an effort to formalize the problem of adversarial examples while using defensive models, Jang et al. [25] suggest a underlying cause for adversarial examples and offer up a solution. Jang et al. proposes that the existence of such adversarial examples stems from the lack of understanding of the topological structure of the underlying data manifold. Simply put, the learned data distribution does not match the true data distribution, and thus there exists images that do not get mapped back to the data manifold by the defense model.

Jang et al. prove a theorem which states that if the latent distribution is composed of $n_Z$ multivariate Gaussian distributions and the data manifold is composed of $n_X$ components, if $n_Z < n_X$ then there exists points outside the data manifold that are part of the approximated data manifold of the latent dimensions. In our proposed model, we use some of the methods they introduced in order endow our model with better topological awareness.

## B Neural Network Architectures

The details of the models used in our paper and experiments are listed in the following three tables.

| **Encoder** | **Decoder** |
|---|---|
| Conv(*, 64, 5, 1, 2) + BN + ReLU | FC(128, 4096) + ReLU |
| Conv(64, 64, 4, 2, 3) + BN + ReLU | ConvT(256, 128, 4, 2, 1) + BN + ReLU |
| Conv(64, 128, 4, 2, 1) + BN + ReLU | ConvT(128, 64, 4, 2, 1) + BN + ReLU |
| Conv(128, 256, 4, 2, 1) + BN + ReLU | ConvT(64, 64, 4, 2, 3) + BN + ReLU |
| FC1(4096, 128), FC2(4096, 128) | ConvT(64, 64, 5, 1, 2) + BN + ReLU |

Table 5: Model structure for Defense-VAE [2]

| Layers | Params |
|---|---|
| Conv.ReLU | $3 \times 3 \times 32$ |
| Conv.ReLU | $3 \times 3 \times 32$ |
| Max Pooling | $2 \times 2$ |
| Conv.ReLU | $3 \times 3 \times 64$ |
| Conv.ReLU | $3 \times 3 \times 64$ |
| Max Pooling | $2 \times 2$ |
| Dense.ReLU | 200 |
| Dense.ReLU | 200 |
| Softmax | 10 |

Table 6: Model structure for classifier [23]

| A | B | C | D, E* |
|---|---|---|---|
| Conv(64, $5 \times 5$, 1) | Dropout(0.2) | Conv(128, $3 \times 3$, 1) | FC(200) |
| ReLU | Conv(64, $8 \times 8$, 2) | ReLU | ReLU |
| Conv(64, $5 \times 5$, 2) | ReLU | Conv(64, $3 \times 3$, 2) | Dropout(0.5) |
| ReLU | Conv(128, $6 \times 6$, 2) | ReLU | FC(200) |
| Dropout(0.25) | ReLU | Dropout(0.25) | ReLU |
| FC(128) | Conv(128, $5 \times 5$, 1) | FC(128) | Dropout(0.5) |
| ReLU | ReLU | ReLU | FC(10) + Softmax |
| Dropout(0.5) | Dropout(0.5) | Dropout(0.5) | |
| FC(10) + Softmax | FC(10) + Softmax | FC(10) + Softmax | |

Table 7: Model structure for Black-Box attack classifier [1]. * indicates same structure without dropout layer.

## C Confusion Matrix of different Models

In sections, we include the confusion matrix of accuracy for other models.

| **true** / **pred** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2418 | 0 | 19 | 2 | 1 | 5 | 14 | 2 | 7 | 1 |
| **1** | 8 | 3379 | 172 | 82 | 58 | 32 | 74 | 154 | 85 | 83 |
| **2** | 19 | 4 | 2566 | 36 | 20 | 5 | 12 | 47 | 62 | 12 |
| **3** | 28 | 9 | 153 | 2618 | 8 | 197 | 20 | 22 | 324 | 30 |
| **4** | 4 | 0 | 28 | 12 | 2483 | 16 | 148 | 41 | 82 | 986 |
| **5** | 125 | 2 | 4 | 120 | 2 | 2141 | 235 | 3 | 177 | 35 |
| **6** | 95 | 4 | 8 | 1 | 22 | 24 | 2316 | 0 | 12 | 4 |
| **7** | 9 | 0 | 110 | 77 | 61 | 7 | 3 | 2600 | 36 | 64 |
| **8** | 112 | 6 | 31 | 38 | 41 | 59 | 52 | 7 | 1866 | 20 |
| **9** | 122 | 1 | 5 | 44 | 250 | 190 | 0 | 208 | 271 | 1792 |
| **Accuracy** | 0.82 | 0.99 | 0.83 | 0.86 | 0.84 | 0.80 | 0.81 | 0.84 | 0.64 | 0.59 |

Table 8: Confusion matrix of accuracy of vanilla Defense-VAE with on entire test datasets (combining all the attacks).

| pred \ true | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2426 | 5 | 23 | 4 | 2 | 12 | 27 | 3 | 23 | 1 |
| **1** | 5 | 3214 | 59 | 40 | 16 | 12 | 22 | 78 | 14 | 35 |
| **2** | 19 | 29 | 2661 | 66 | 21 | 5 | 11 | 47 | 60 | 12 |
| **3** | 22 | 14 | 121 | 2615 | 2 | 211 | 17 | 46 | 245 | 29 |
| **4** | 8 | 29 | 33 | 3 | 2279 | 11 | 101 | 23 | 85 | 704 |
| **5** | 65 | 5 | 3 | 95 | 5 | 1933 | 109 | 3 | 82 | 22 |
| **6** | 91 | 11 | 17 | 2 | 26 | 27 | 2465 | 0 | 26 | 0 |
| **7** | 25 | 25 | 119 | 64 | 90 | 15 | 3 | 2583 | 21 | 155 |
| **8** | 85 | 21 | 47 | 76 | 48 | 151 | 106 | 7 | 2154 | 66 |
| **9** | 194 | 52 | 13 | 65 | 457 | 299 | 13 | 294 | 212 | 2003 |
| **Accuracy** | 0.83 | 0.94 | 0.86 | 0.86 | 0.77 | 0.72 | 0.86 | 0.84 | 0.74 | 0.66 |

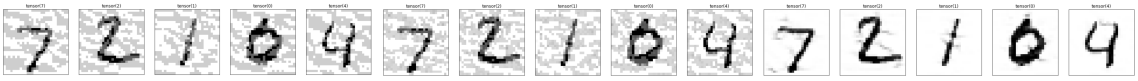
Table 9: Confusion matrix of accuracy of Defense-VAE with classification loss on entire test datasets (combining all the attacks).

| pred \ true | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 2041 | 0 | 15 | 1 | 1 | 6 | 24 | 1 | 5 | 2 |
| **1** | 13 | 3298 | 133 | 25 | 54 | 15 | 35 | 88 | 29 | 43 |
| **2** | 65 | 4 | 2290 | 30 | 40 | 9 | 21 | 48 | 63 | 12 |
| **3** | 138 | 18 | 354 | 2655 | 46 | 283 | 73 | 117 | 497 | 82 |
| **4** | 5 | 8 | 20 | 6 | 2165 | 12 | 131 | 31 | 63 | 869 |
| **5** | 212 | 6 | 12 | 183 | 16 | 2159 | 320 | 7 | 169 | 50 |
| **6** | 57 | 9 | 9 | 0 | 20 | 25 | 2115 | 0 | 14 | 2 |
| **7** | 74 | 32 | 224 | 55 | 201 | 15 | 7 | 2615 | 34 | 266 |
| **8** | 53 | 25 | 32 | 40 | 74 | 57 | 145 | 11 | 1875 | 72 |
| **9** | 282 | 5 | 7 | 35 | 329 | 95 | 3 | 166 | 173 | 1629 |
| **Accuracy** | 0.69 | 0.97 | 0.74 | 0.88 | 0.73 | 0.81 | 0.74 | 0.85 | 0.64 | 0.54 |

Table 10: Confusion matrix of accuracy of Defense-VAE with combined loss on entire test datasets (combining all the attacks).

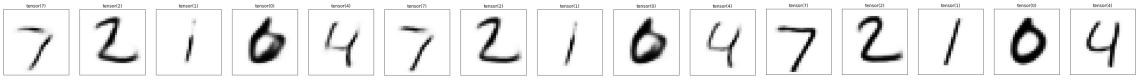
## D Adversarial Images and Model Outputs

In this section, we provide the adversarial images and out models' outputs.



(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 7: Adversarial examples.



(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 8: Output images of vanilla model based on the corresponding adversarial inputs.

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 9: Output images of model trained with classification loss function based on the corresponding adversarial inputs.

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

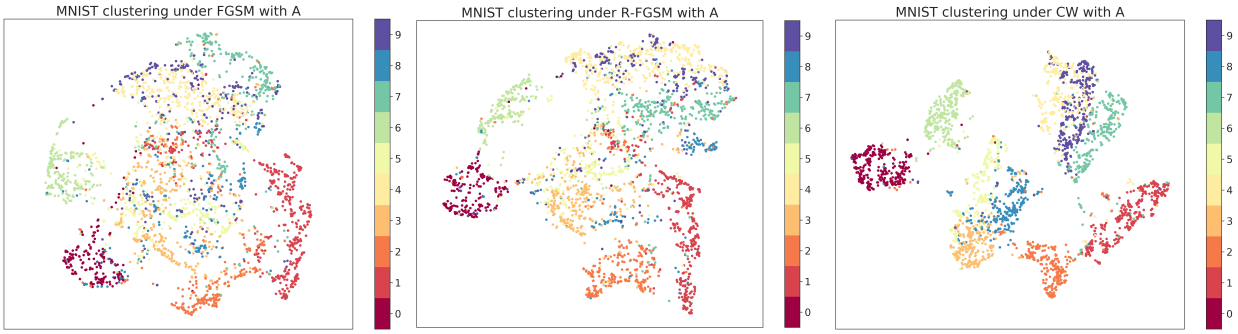
Figure 10: Output images of model trained on proximity and distance loss based on the corresponding adversarial inputs.

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 11: Output images of model trained on combined loss based on the corresponding adversarial inputs.

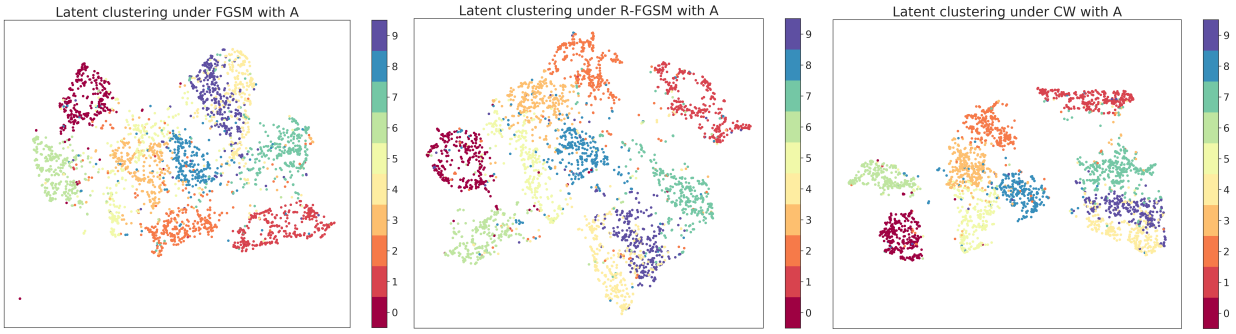
## E Additional Plots for Experiments

In this section, we will provide the clustering plots of model output and latent variables that have not been included in the sections above.



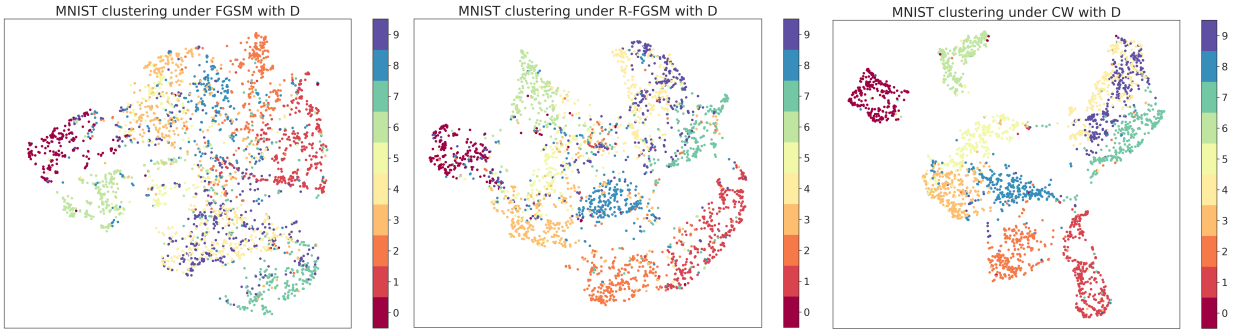

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 12: Clustering plots of output from vanilla model across various adversarial inputs.



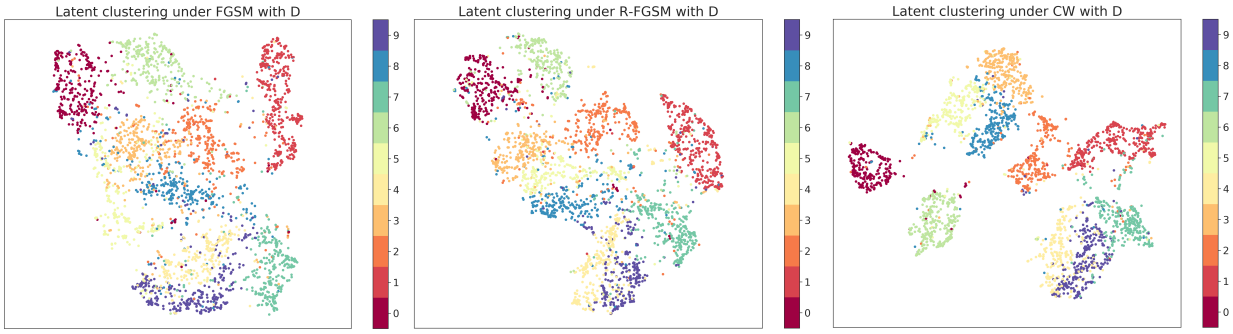

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 13: Clustering plots for latent variables of the vanilla model given adversarial examples as input.



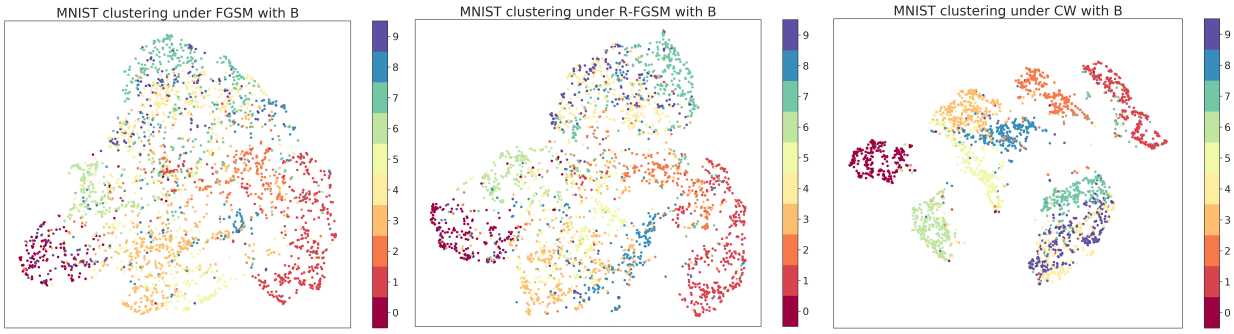

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 14: Clustering plots of output from model trained with classification loss across various adversarial inputs.



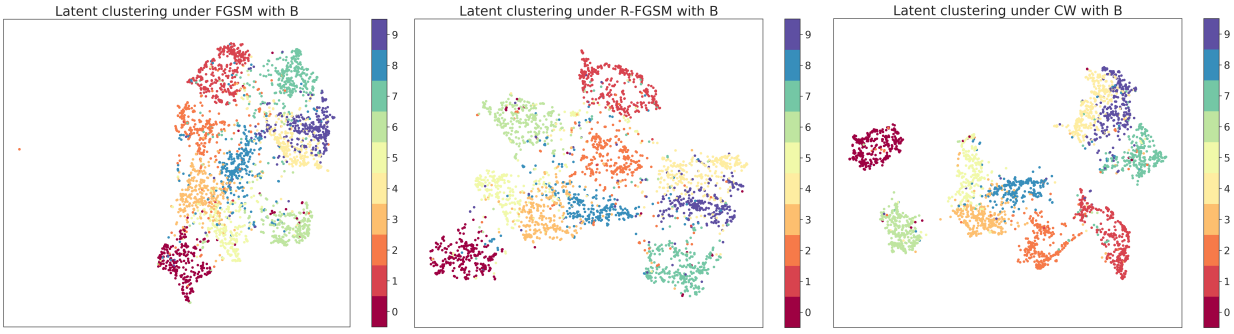

(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 15: Clustering plots for latent variables of the model trained with classification loss given adversarial examples as input.




(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 16: Clustering plots of output from model trained with combined loss across various adversarial inputs.




(a) FGSM attack. (b) Rand-FGSM attacks (c) CW attack.

Figure 17: Clustering plots for latent variables of the model trained with combined loss given adversarial examples as input.