Towards Robustness of Deep Neural Networks via Regularization

Yao Li\textsuperscript{1} Martin Renqiang Min\textsuperscript{2} Thomas Lee\textsuperscript{3} Wenchao Yu\textsuperscript{2} Erik Kruus\textsuperscript{2} Wei Wang\textsuperscript{4} Cho-Jui Hsieh\textsuperscript{4} \\
\textsuperscript{1}University of North Carolina, Chapel Hill \textsuperscript{2}NEC Labs America, Princeton \textsuperscript{3}University of California, Davis \textsuperscript{4}University of California, Los Angeles

Abstract

Recent studies have demonstrated the vulnerability of deep neural networks against adversarial examples. Inspired by the observation that adversarial examples often lie outside the natural image data manifold and the intrinsic dimension of image data is much smaller than its pixel space dimension, we propose to embed high-dimensional input images into a low-dimensional space and apply regularization on the embedding space to push the adversarial examples back to the manifold. The proposed framework is called Embedding Regularized Classifier (ER-Classifier), which improves the adversarial robustness of the classifier through embedding regularization. Besides improving classification accuracy against adversarial examples, the framework can be combined with detection methods to detect adversarial examples. Experimental results on several benchmark datasets show that, our proposed framework achieves good performance against strong adversarial attack methods.

1. Introduction

It has been shown that Deep Neural Networks (DNNs) are vulnerable to adversarial examples that are generated by adding carefully crafted perturbations to original images [48, 30]. This phenomenon brings out security concerns for practical applications of deep learning. Despite many adversarial training methods have been proposed for defending against adversarial examples [38, 55, 41, 42], we propose a novel defense and detection method from a different point of view.

Recent work showed that adversarial examples often lie outside the natural image data manifolds [27, 49]. Since the model was trained using data in the manifold, the model naturally mis-classifies on examples outside the manifold. Therefore, applying regularization to push the adversarial examples back to the natural image data manifold may help improve the robustness of neural networks (see Figure 1). Furthermore, a consensus in the high-dimensional data analysis community is that, a method working well on the high-dimensional data is because the data is not really of high-dimension [33]. Inspired by the observation that the intrinsic dimension of image data is actually much smaller than its pixel space dimension [33] and adversarial examples lie outside the natural image data manifold [27, 49], we propose a defense framework called Embedding Regularized Classifier (ER-Classifier) shown in Figure 2.

The difference between ER-Classifier and general deep classifier is that the extracted feature is regularized by a discriminator part in ER-Classifier. To be more specific, we introduce a discriminator in the latent space which tries to separate the generated code vectors (output of the encoder network) from the ideal code vectors (sampled from a prior distribution, i.e., a standard Gaussian distribution). Employing a similar powerful competitive mechanism as demonstrated by Generative Adversarial Networks [20], the discriminator enforces the embedding space of the model to follow the prior distribution. This regularization process can help remove the effect of adversarial distortion and push the adversarial example back to the natural data manifold. Another difference is that the embedding space dimension of ER-classifier is much smaller, which makes it easier for ER-Classifier to apply regularization.

We compare ER-Classifier with other state-of-the-art defense methods on MNIST, CIFAR10, STL10 and Tiny Imagenet. Experimental results demonstrate that our proposed ER-Classifier outperforms other methods by a large margin. To sum up, this paper makes the following four main contributions:

- A novel unified end-to-end robust deep neural net-
work framework against adversarial attacks is proposed, where embedding space regularization is applied to remove the adversarial effect.

- An objective is induced to minimize the optimal transport cost between the true class distribution and the framework output distribution, guiding the framework to project the input image to a low-dimensional space without losing important features for classification.
- A detection process is proposed to further improve the robustness of the framework.
- Extensive experiments demonstrate the robustness of our proposed ER-Classifier framework under the white-box attacks, and show that ER-Classifier outperforms other state-of-the-art approaches on several benchmark image datasets.

2. Related Work

In this section, we summarize related work into two categories: attack methods and defense mechanisms.

Attack Methods. We first discuss different adversarial attack methods [38, 6, 24, 13, 21, 5, 9, 43, 12, 24, 44, 58, 56]. Two main types of attack settings have been considered in previous research: black-box and white-box settings. Under the white-box setting, attackers have all information about the targeted neural network, including network structure and gradients. Many other white-box attack methods have been proposed [38, 6, 24, 13], and among them C&W [6] and PGD [38] attacks have been widely used to test the robustness of machine learning models. Recently, a new attack method called Autoattack [13] was developed, which is parameter-free, computationally affordable and user-independent, to test adversarial robustness. In this paper, we mainly use $l_\infty$-PGD untargeted attack [38] and Autoattack [13] to evaluate the effectiveness of the defense method under the white-box setting. In the black-box setting, the detailed model information, such as gradient and model structure, are not available to the attackers. Some attack methods rely on the predicted scores, such as class probabilities or logits, of the model to craft adversarial ex-

amples [9, 25]. Some attacks are more agnostic and only rely on the final decision of the model [3, 10, 23, 8, 35, 7].

Defense Mechanisms. Many works have been done to improve the robustness of deep neural networks [55, 38, 40, 21, 34, 52, 16, 53, 54, 28, 45, 22, 4, 32, 57]. We select five representative methods to compare with the proposed framework. Madry’s adversarial training [38] proposed a min-max formulation against adversarial attacks. The proposed model is trained on adversarial examples generated during the training process within the $c$-ball of input images. A new loss function was introduced in [55] to trade adversarial robustness off classification accuracy. The loss function is used to replace the loss function used in Mary’s adversarial training to further improve the robustness of adversarial training. Another effective defense method under the white-box setting is RSE [34]. The authors proposed a “noise layer”, which fuses output of each layer with Gaussian noise. Due to the difficulty of defense, some works attempted to detect adversarial examples as alternative solutions. In [19], the author proposed a kernel density based detection framework, in which one kernel density model is fitted for each class on the final layer output. A Logistic Regression model is trained on the kernel density scores to detect adversarial examples. Local Intrinsic Dimensionality (LID) characterizes the space-filling capability of the region surrounding a sample, based on the distance to its nearest neighbors within the sample. The author of [36] proposed to use LID as a feature to facilitate the detection of adversarial examples.

Notations. In this paper, we use $l_\infty$ and $l_2$ distortion metrics to measure similarity. We report $l_\infty$ distance in the normalized $[0, 1]$ space, so that a distortion of 0.031 corresponds to $8/256$, and $l_2$ distance as the total root-mean-square distortion normalized by the total number of pixels. We use calligraphic letters for sets (i.e., $\mathcal{X}$), capital letters for random variables (i.e., $X$), and lower case letters for their values (i.e., $x$). The probability distributions are denoted with capital letters (i.e., $P_X$) and corresponding densities with lower case letters (i.e., $p_X$).

3. Embedding Regularized Classifier

3.1. Framework Details

We propose a novel defense framework, ER-Classifier, which aims at improving the robustness of deep neural networks through embedding regularization by training a discriminator to push the embedding space distribution towards a prior distribution. An overview of the framework is shown in Figure 2.

Mathematically, input images $X \in \mathcal{X} = \mathbb{R}^d$ from $P_X$ are projected to a low-dimensional embedding vector $Z \in Z = \mathbb{R}^k$ through the encoder $Q_{\psi}$, where the embedding space dimension $k$ is much smaller than the pixel
space dimension $d$. The discriminator $D_{\gamma}$ discriminates between the generated code $\tilde{Z} \sim Q_{\phi}(Z|X)$ and the ideal code $Z \sim P_{Z}$. In this paper we apply the standard Gaussian $N(0, 1)$ as our prior distribution $P_{Z}$, but other priors may be used for different cases. The classifier $C_{\tau}$ performs classification based on the generated code $\tilde{Z}$, producing output $U \in \mathcal{U} = \mathbb{R}^{m}$, where $m$ is the number of classes. The label of $X$ is denoted as $Y \in \mathcal{U}$.

The training objective of ER-Classifier is:

$$\inf_{Q(z|x) \in Q} \mathbb{E}_{P_{X}} \mathbb{E}_{Q(z|x)} \{ \ell(Y, C(Z)) \} + \lambda \mathcal{D}(Q_{\phi}, P_{Z}).$$

The first part of the objective function represents classification loss, where the encoder $Q$ maps the input image to the embedding vector $Z$, then the classifier $C$ takes $Z$ as an input and performs prediction. $Y$ is the true label of the input image and $\ell$ is the cost function. The second part represents the regularisation loss, where $\lambda > 0$ is a hyper-parameter controls the trade-off between regularisation and classification tasks, and $\mathcal{D}$ can be any arbitrary divergence between the marginal distribution of $Z$ ($Q_{\phi}$) and prior distribution ($P_{Z}$).

To estimate the divergence between $Q_{\phi}$ and $P_{Z}$, we apply a GAN-based framework, fitting a discriminator to estimate the 1-Wasserstein distance between $Q_{\phi}$ and $P_{Z}$:

$$W(Q_{\phi}, P_{Z}) = \inf_{\gamma \in \Pi(Q_{\phi}, P_{Z})} \mathbb{E}_{(\tilde{Z}, Z) \sim \Gamma} \| \tilde{Z} - Z \|.$$

When training the framework, the weight clipping method proposed in Wasserstein GAN [1] is applied to help stabilize the training of discriminator $D_{\gamma}$. Details of the training algorithm is summarized in Algorithm 1.

At training stage, the encoder $Q_{\phi}$ first maps the input $x$ to a low-dimensional space, resulting in generated code ($\tilde{z}$). Another ideal code ($z$) is sampled from the prior distribution, and the discriminator $D_{\gamma}$ discriminates between the ideal code and the generated code. The classifier ($C_{\tau}$) predicts the image label based on the generated code ($\tilde{z}$). The main goal of ER-Classifier is leveraging embedding space regularization to push adversarial examples back to the natural data manifold, removing adversarial perturbation. Therefore, other defense methods can also benefit from this property. Our framework is trained with min-max robust optimization [38], first searching for adversarial examples with projected gradient descent (PGD) method then optimizing the framework over the adversarial examples.

At inference time, only the encoder $Q_{\phi}$ and the classifier $C_{\tau}$ are used. The input image $x$ is first mapped to a low-dimensional space by the encoder ($\tilde{z} = Q_{\phi}(x)$), then the latent code $\tilde{z}$ is fed into the classifier to predict the label.

### 3.2. Justifications of the Framework

In the framework, the classifier $(P_{C}(U|Z))$ maps a latent code $\tilde{Z}$ sampled from a fixed distribution ($P_{Z}$) in a latent space $Z$, to the output $U \in \mathcal{U} = \mathbb{R}^{m}$. The density of ER-Classifier output is defined as follow:

$$p_{C}(u) := \int_{z} p_{C}(u|z)p_{Z}(z)dz, \forall u \in \mathcal{U}.$$  

(2)

If the divergence between the distribution of the true class ($P_{Y}$) and the distribution of the framework output ($P_{C}$) is minimized, the framework will be doing classification task well, which is the most important goal of the framework.

There are various ways to define the distance or divergence between $P_{Y}$ and $P_{C}$. In this paper, we turn to the optimal transport theory [51], because it imposes a weak distance between distributions making it easier for a sequence of distributions to converge [1]. Kantorovich’s distance induced by the optimal transport problem is given by

$$W_{c}(P_{Y}, P_{C}) := \inf_{\gamma \in \mathcal{P}(Y \times \mathcal{U})} \mathbb{E}_{(Y,U) \sim \mathcal{P}} \{ c(Y,U) \},$$

where $\mathcal{P}(Y \sim P_{Y}, U \sim P_{C})$ is the set of all joint distributions of $(Y,U)$ with marginals $P_{Y}$ and $P_{C}$, and $c(y,u) : \mathcal{U} \times \mathcal{U} \rightarrow \mathbb{R}_{+}$ is any measurable cost function. $W_{c}(P_{Y}, P_{C})$ measures the divergence between probability distributions $P_{Y}$ and $P_{C}$. When the probability measures are on a metric space, the $p$-th root of $W_{c}$ is called the $p$-Wasserstein distance.

### Algorithm 1: Training ER-Classifier

1. **Input**: Regularization coefficient $\lambda > 0$, $l_{\infty}$ distortion $\epsilon$, step-size for PGD attack $\alpha$.
2. **Note**: $\ell$ stands for the cross-entropy loss. $\Pi_{\epsilon}(\cdot, x^{o})$ stands for projection to the set $\{ x \| x - x^{o} \|_{\infty} \leq \epsilon \}$
3. **while** $\langle \phi, \gamma, \tau \rangle$ not converged **do**
   4. Sample $\{ (x_{1}^{o}, y_{1}), ..., (x_{n}^{o}, y_{n}) \}$ from the training set. Generate adversarial examples by running PGD for multiple steps, and get $\{ (x_{1}, y_{1}), ..., (x_{n}, y_{n}) \}$
   5. Sample $\{ z_{1}, ..., z_{n} \}$ from the prior $P_{Z}$
   6. Obtain $\tilde{z}_{i}$ from $Q_{\phi}(z|x_{i})$ for $i = 1, ..., n$
   7. **Update** $D_{\gamma}$ by ascending:
      \[
      \frac{\lambda}{n} \sum_{i=1}^{n} D_{\gamma}(z_{i}) - D_{\gamma}(\tilde{z}_{i})
      \]
   8. **Update** $Q_{\phi}$ and $C_{\tau}$ by descending:
      \[
      \frac{1}{n} \sum_{i=1}^{n} \ell(C_{\tau}(Q_{\phi}(x_{i})), y_{i})
      \]
   9. **Update** $Q_{\phi}$ by ascending the following objective by 1-step Adam:
      \[
      \frac{\lambda}{n} \sum_{i=1}^{n} D_{\gamma}(Q_{\phi}(x_{i}))
      \]
Theorem 1 For \( P_C \) as defined above with a deterministic \( P_C(U|Z) \) and any function \( C : Z \mapsto U \),

\[
W_c(P_Y, P_C) = \inf_{Q,Z|P_Z} \mathbb{E}_{P_X} \mathbb{E}_{Q(Z|X)} \{ c(Y, C(Z)) \},
\]

where \( c(y,u) : U \times U \mapsto \mathbb{R}_+ \) is any measurable cost function. \( Q_Z \) is the marginal distribution of \( Z \) when \( X \sim P_X \) and \( Z \sim Q(Z|X) \). The proof is presented in Appendix A.

It is obvious that the objective function (1) is relaxed from the r.h.s. of Theorem 1, by converting the constraint on \( Q_Z \) to a penalty term and using cross-entropy loss as the cost function. Therefore, optimizing over (1) is equivalent to minimizing the discrepancy between the true class distribution \( (P_Y) \) and the output distribution \( P_C \). A summary of Theorem 1 is that if we are doing optimization well on the objective used, we will be doing classification task well. Theorem 1 requires a deterministic \( P_C(U|Z) \). However, our proposed framework readily applies to the non-deterministic case. We derived an upper bound on the distance between the two distributions \( P_Y \) and \( P_C \) for the non-deterministic case. See details in Appendix A.

3.3. Detecting Adversarial Examples

Studies showed that adversarial samples come from a distribution that is different from the natural data distribution, that is, adversarial samples do not lie on the data manifold, and DNNs perform correctly only near the manifold of training data [27, 49]. Embedding regularization helps push adversarial examples back to the data manifold but there might be some adversarial examples that are hard to regularize. Therefore, we propose a detection framework, ER-Detector, to filter out these adversarial examples before classification. After training the ER-Classifier framework, all samples from the training set are fed into the encoder to generate the combined code \( Z' \), where \( Z' \) is generated by concatenating the hidden layer output means of encoder \( Q_\phi \) and encoder direct output \( \hat{Z} \). Combined code is used to provide more clues to the detector. Then, one kernel density model is fitted for each class based on \( Z' \).

With the kernel density model, we can get density score for any input. A logistic regression model \( (G_\beta) \) is trained on the density scores to detect adversarial examples, with scores of adversarial examples as positive examples and scores of natural examples as negative examples. Details of the detector training process are shown in Algorithm 2 in Appendix B.

At the inference time, only the encoder \( Q_\phi \), classifier \( C_\tau \) and detector \( G_\beta \) are used. The input image \( X \) is first mapped to a low-dimensional space \( (Q_\phi(X) \rightarrow \hat{Z}, Z') \). Then the embedding based detector \( G_\beta \) will judge whether the image is an adversarial example. If it is marked as adversarial, the classifier does not need to deal with it, otherwise the classifier will predict the label based on the latent code. The process is designed to detect adversarial examples that manage to “escape” the regularization process and further improve the robustness of the framework.

4. Experiments

For the first set of experiments, we assume the classifier needs to predict a label for each test sample and compare the performance of the proposed algorithm (ER-Classifier) with other state-of-the-art adversarial defense methods. We consider the following datasets: MNIST [31], CIFAR10 [29], STL10 [11] and Tiny Imagenet [14]. See details of data in Appendix C.

Various defense methods have been proposed to improve the robustness of deep neural networks. In Section 4.1, we compare our algorithm with state-of-the-art methods that are robust in the white-box setting. Madry’s adversarial training (Madry’s Adv) has been recognized as one of the most successful defense methods in the white-box setting, as shown by [2]. Random Self-Ensemble (RSE) method introduced by [34] adds stochastic components in the neural network, achieving similar performance to Madry’s adversarial training algorithm. Trades introduced in [55] won first place in the NeurIPS 2018 Adversarial Vision Challenge and outperformed the runner-up apparoach by 11.41% in terms of mean \( l_2 \) perturbation distance.

Since the main goal of ER-Classifier is using embedding regularization to improve adversarial robustness, other defense methods can also benefit from this property. The proposed ER-Classifier is trained with min-max robust optimization [38]. ER-Trades is a variant that combines the proposed framework with the loss function of Trades [55]. To demonstrate the regularization effect of ER-Classifier, we include a variant ER-Classifier which trains ER-Classifier without min-max robust optimization. Code for reproduction is available in supplementary material and will be made available at Github later, and network architecture details are included in Appendix G.

In Section 4.5, ER-Detector, ER-Classifier combined with detection method described in Section 3.3, is compared with KD-Detection [19] and LID Detection [36] on MNIST [31] and CIFAR10 [29].

4.1. Evaluate Models Under White-box Attack

In this section, we first evaluate the defense methods against \( l_\infty \)-PGD untargeted attack [38]. The methods that perform best among the baselines are then evaluated by Autoattack [13]. When tested against PGD attack, defense methods are evaluated under different distortion levels (\( \epsilon \)), and the larger the distortion the stronger the attack. Depending on the image scale and type, different datasets are sensitive to different strengths of the attack. Models on MNIST are evaluated under distortion level from 0 to 0.4 by
Based on Figure 3, we can see that ER-Classifier is the most robust one on MNIST and Tiny Imagenet. On STL10, ER-Trades outperforms all the other baselines. On CIFAR10, ER-Classifier- performs better than other baselines when the attack is strong and other baselines except that Trades performs similarly when distortion level ($\epsilon$) is small. The good performances of ER-Classifier and ER-Trades show that the proposed framework can be combined with state-of-the-art defense methods to further improve robustness against adversarial examples.

ER-Classifier without min-max robust optimization can also improve the robustness of deep neural networks. Compare the performance of ER-Classifier$^-$ with the performance of the model without defense method (No Defense), we can see that ER-Classifier$^-$ is much more robust than the model with no defense method on all benchmark datasets. Besides, when the distortion level ($\epsilon$) is large, ER-Classifier$^-$ tends to perform better than some state-of-the-art defense methods on MNIST, CIFAR10 and Tiny Imagenet. This phenomenon is obvious on CIFAR10 and it even performs better than ER-Classifier when the attack strength is strong. The reason might be that without min-max robust optimization, it is easier to regularize the embedding space.

Madry’s adversarial training performs best among all the baselines except proposed frameworks. Therefore, we evaluate Madry’s adversarial training and ER-Classifier against a stronger white-box attack, Autoattack [13]. Since Autoattack takes a long time to generate adversarial examples, the two methods are compared on 1,000 random samples from each benchmark dataset. The results are shown in Table 1.

Table 1. Testing accuracy (%) of two defense methods under Autoattack [13] with $l_\infty$ norm. VGG19 is used as the base architecture on CIFAR10. Architectures on STL10 and Tiny Imagenet are similar to VGG19. See details of architectures in Appendix G.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>STL10</th>
<th>Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>Madry’s Adv</td>
<td>69.0</td>
<td>35.6</td>
<td>21.0</td>
<td>9.70</td>
</tr>
<tr>
<td>ER-Classifier</td>
<td>79.0</td>
<td>47.2</td>
<td>25.3</td>
<td>11.1</td>
</tr>
</tbody>
</table>

0.025. Models on CIFAR10 and STL10 are evaluated under $\epsilon \in [0, 0.06, 0.005]$. Models on Tiny Imagenet are evaluated under $\epsilon \in [0, 0.02, 0.002]$. When evaluated by Autoattack, the distortion levels on MNIST, CIFAR10, STL10 and Tiny Imagenet are 0.3, 0.03, 0.03 and 0.01 respectively. As mentioned in the notation part, all the distortion levels are reported in the normalized $[0, 1]$ space. All the methods are trained for 30 epochs on the datasets. The experimental results against $l_\infty$-PGD are shown in Figure 3.

We also compare Defense-GAN [46] with our method ER-Classifier on MNIST. Although Defense-GAN was shown to be partly broken by [2, 26], both ER-Classifier and Defense-GAN leverage the power of generative models to improve adversarial robustness, and comparing to Defense-GAN is important to demonstrate the advantage of our novel Wassserstein distance regularization. Please note that Defense-GAN is not our major comparison baseline in this paper.

Both ER-Classifier and Defense-GAN are evaluated against the $l_2$-C&W untargeted attack, one of the strongest white-box attacks proposed by [6]. Defense-GAN is evaluated using the method proposed by [2], and the code is available on github. ER-Classifier is evaluated against $l_2$-C&W untargeted attack with the same hyper-parameter values as those used in the evaluation of Defense-GAN. The results under $l_2 \leq 0.005$ threshold are shown in Table 2. Based on Table 2, ER-Classifier is much more robust than Defense-GAN under the $l_2 \leq 0.005$ threshold. Since [46] did not evaluate Defense-GAN on CIFAR10, STL10 and Tiny Imagenet, without details of GAN structure, we can not compare with Defense-GAN on these datasets.

### 4.2. Evaluate Models Under Black-box Attack

We evaluate ER-Classifier against a recently proposed black-box attack method called RayS [7] on four benchmark datasets to test the performance of the proposed framework under black-box setting. RayS is an adversarial attack that only requires the target model’s prediction. RayS is performed on 1,000 random samples from each benchmark dataset since the attack process takes a long time. In the experiment, the maximum number of queries is set to be 10,000. We report the robust accuracy in Table 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>CIFAR10</th>
<th>STL10</th>
<th>Tiny</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER-Classifier</td>
<td>75.8</td>
<td>51.0</td>
<td>34.4</td>
<td>21.7</td>
</tr>
</tbody>
</table>

4.3. Evaluate the Effect of Discriminator

The ER-Classifier framework consists of three parts, where the classification task is done by the encoder $Q_{\phi}$ and classifier $C_{\gamma}$, and the regularization task is done by the discriminator $D_{\omega}$. The encoder and classifier are not different from the general deep neural network classifier. To show that it is embedding space regularization improves the robustness, we fit a framework with only the encoder and classifier part (E-CLA), where the encoder and classifier have the same structures as in ER-Classifier, and com-

1Publicly available at https://github.com/anishathalye/obfuscated-gradients/tree/master/defensegan
2Code available at https://github.com/uclaml/RayS
compare E-CLA with ER-Classifier framework. For a fair comparison, both structures are trained without min-max robust optimization. The results are shown in Figure 4.

Based on Figure 4, we can observe that ER-Classifier is much more robust than E-CLA structure on MNIST, CIFAR10 and Tiny Imagenet. It is also more robust on STL10 but not that much. The reason might be that there are only 5,000 training images in STL10 and the resolution is 96 × 96. Therefore, it is harder to learn a good embedding with a limited amount of images. However, even when the number of training images is limited, ER-Classifier is still much more robust than the E-CLA structure. This observation demonstrates that regularization on the embedding space helps improve the adversarial robustness. Notice that the performance of the E-CLA structure is similar to the performance of the model without defense method on CIFAR10, STL10 and Tiny Imagenet, and worse on MNIST, which means the robustness of ER-Classifier does not come from the structure design.

Variational auto-encoder can project the images to low-dimensional space and use Kullback–Leibler divergence loss to regularize the embedding distribution, which does not need discriminator structure. Therefore, we also tried VAE-CLA, which applies Variational auto-encoder structure to do the projection and regularization. The experimental results in Figure 4 show that VAE-CLA does not perform as well as ER-Classifier. Based on the observation of the Kullback–Leibler loss and classification loss during the training process, it seems difficult for VAE-CLA to balance between the two tasks.

4.4. Embedding Visualization

In this section, we compare the embedding learned by Encoder+Classifier structure (E-CLA) and the embedding learned by ER-Classifier without min-max robust optimization on several datasets. We first generate embedding of testing data using the encoder (z = Q_{ϕ}(x)), then project the embedding (z) to 2-D space by tSNE [37]. Adversarial images (x_{adv}) are generated using l_{∞}-PGD attack. The adversarial embedding is generated by feeding the adversarial images into the encoder (z_{adv} = Q_{ϕ}(x_{adv})). Finally, we project the adversarial embedding (z_{adv}) to 2-D space. The results are shown in Figures 5 and 6. The first two plots are embedding visualization for E-CLA, and the last two plots are the embedding visualization for ER-Classifier. In adversarial embedding visualization plots, the mis-classified point is marked as “down triangle”, which means the PGD attack successfully changed the prediction, and the correctly classified point is marked as “point”, which means the attack fails.

We can see that E-CLA can learn a good embedding on natural images of MNIST. In the first plot of Figure 5, embedding for different classes are well separated on the 2D space, but under adversarial attack (second plot of Figure 5), some points of different classes are mixed together. However, ER-Classifier can generate good separated embedding on both natural and adversarial images (last two plots of Figure 5). On CIFAR10, the E-CLA can not generate good separated embedding on either natural or adversarial images (first two plots of Figure 6), while ER-Classifier can generate good separated embedding for both (last two plots of Figure 6).

4.5. ER-Detector

In this section, we compare the performance of the proposed detection method (ER-Detector) with baseline adversarial example detection methods on MNIST and CIFAR10 to show that the ER-Classifier framework can be combined with a detection method to further improve the adversarial robustness. There are two versions of ER-Detector: ER1 performs classification on the direct output Z and ER2 performs classification on the combined code Z'.

4.5.1 Setup and Criteria

For both MNIST and CIFAR10, deep neural networks without any defense methods are used as baseline nets for KD-Detection and LID detection methods. The networks are trained on the designated training set. The designated testing set is split into set I (20%) and set II (80%). The detectors for KD, LID and ER-Detector, are trained on set I, then evaluated on set II (80%).

The detection methods are evaluated against FGSM, PGD and C&W attacks, which are frequently used to benchmark the detection methods. For FGSM and PGD, the l_{∞} distortion levels (e) are 0.3 on MNIST and 0.03 on CIFAR10. Those are standard values used in many previous
For $\epsilon$-PGD attack on four different datasets: MNIST, CIFAR10, STL10 and Tiny Imagenet.

Figure 4. Testing accuracy of E-CLA, VAE-CLA and ER-Classifier on MNIST, CIFAR10, STL10 and Tiny Imagenet.

Figure 5. 2D embeddings for E-CLA and ER-Classifier on MNIST. Larger visualization figures are available in Appendix E.

Figure 6. 2D embeddings for E-CLA and ER-Classifier on CIFAR10. Larger visualization figures are available in Appendix E.

Table 4. Criteria for evaluating the detection methods

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Description</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC-NADV</td>
<td>detecting accuracy for non-adversarial examples</td>
<td>ACC-NADV</td>
</tr>
<tr>
<td>ACC-CLA</td>
<td>classification accuracy for adversarial examples</td>
<td>ACC-CLA</td>
</tr>
<tr>
<td>ACC-COM</td>
<td>combined accuracy for adversarial example</td>
<td>ACC-COM</td>
</tr>
</tbody>
</table>

The detection method should not detect the non-adversarial examples as adversarial ones, i.e. the false positive rate should not be high. The higher the ACC-NADV, the lower the false positive rate. Besides, the detection method should also recognize the adversarial examples correctly, and if not detected, it should classify them correctly. Since ACC-ADV only considers the detection accuracy of adversarial examples, it does not fully represent the “ability” of detectors in handling adversarial examples. Instead, ACC-COM represents the accuracy for detecting and classifying adversarial examples. Therefore, ACC-NADV and ACC-COM are two important criteria reflecting the overall performance of the detectors.

4.5.2 Detectors Against “Familiar” Adversary

In this part, we test the performance of the detection methods against the adversary that have been “seen” by the detectors, which means that the detectors are first trained with the adversarial examples generated by an attack method, and then tested with adversarial examples crafted by the same attack method. The performance of the detection methods are shown in Table 5. The numbers in the table are percentage (%), and the best ACC-NADV and ACC-COM are marked in bold.

Based on Table 5, ERs perform better than KD and LID on MNIST, especially when evaluated against the PGD attacks. KD also performs well when tested against the C&W attack, but the accuracy for detecting non-adversarial examples is lower than those of ERs. ERs also perform better...
4.5.3 Cross Attack

We now evaluate the performances of the detectors under the setting that the detectors are trained on adversarial examples generated by one attack method, and tested on adversarial examples crafted by another attack method. This is a more realistic setting since the detection system cannot predict what kind of attack strategy will be used by the adversary. Therefore, the performances of detection methods under the cross-attack setting are important. The experimental results are shown in Tables 6-8 (“adv.examples” stands for “adversarial examples”).

From Tables 6-8, ERs perform better than KD and LID on MNIST in cross-attack setting, especially when trained on PGD adversarial examples and attacked by C&W. When trained on PGD adversarial examples and attacked by FGSM, KD performs better in terms of ACC-NADV. However, the corresponding ACC-COM of KD is much lower than those of ERs. The ACC-COM of KD is only 10.06% while the ACC-COMs of ERs are 100%. Taking both criteria into consideration, ERs perform better than KD and LID when trained on FGSM adversarial examples and attacked by other methods on CIFAR10. From Table 7, when trained on PGD adversarial examples and attacked by FGSM on CIFAR10, ERs perform better than KD and LID. However, when attacked by C&W, KD performs the best and ERs perform slightly worse in terms of ACC-NADV. Based on Table 8, when trained on C&W adversarial examples and attacked by PGD on CIFAR10, ERs perform better than other baseline methods. When attacked by FGSM, ACC-COM of KD is slightly better than those of ERs. The difference is not significant as the ACC-COM of ER2 is 91.18% and that of KD is 90.58%. What's more, ACC-NADV of KD is much lower than those of ERs. The ACC-COM of KD is much lower than those of ERs. Therefore, taking both criteria into consideration, ER2 performs the best.

See more experimental results on adversarial detection in Appendix D. ER-Detector also performs well against high-confidence adversarial examples. Due to the page limit, the results of comparing detectors against high confidence adversarial examples and the results of evaluating the effect of regularization on detection performance are moved to Appendix D.

5. Conclusion

In this paper, we propose a new defense framework, ER-Classifier, which improves the robustness of deep neural networks through embedding regularization. A discriminator is trained to minimize the discrepancy between the embedding space distribution and the prior distribution. Theoretical analysis shows that our framework is not distracted from the main goal of the model, to do classification well. We empirically show that ER-Classifier is more robust than other state-of-the-art defense methods on several benchmark datasets. Future work will include further exploration of the low-dimensional space to improve the robustness of deep neural networks.

Acknowledgements This work is partially supported by NSF under CCF-1934568, DMS-1916125, DMS-2113605, IIS-1901527, IIS-2008173 and IIS-2048280.
References


A. Theoretical analysis of our framework

The proof of Theorem 1 is adapted from the proof of Theorem 1 in [50]. Consider certain sets of joint probability distributions of three random variables \((X, U, Z) \in X \times U \times Z\). \(X\) can be taken as the input images, \(U\) as the output of the framework, and \(Z\) as the latent codes. \(P_{C, z}(U, Z)\) represents a joint distribution of a variable pair \((U, Z)\), where \(Z\) is first sampled from \(P_Z\) and then \(U\) from \(P_C(U|Z)\). \(P_C\) defined in (2) is the marginal distribution of \(U\) when \((U, Z) \sim P_{C, z}\).

The joint distributions \(\Gamma(X, U)\) or couplings between values of \(X\) and \(U\) can be written as \(\Gamma(X, U) = \Gamma(U|X)P_X(X)\) due to the marginal constraint. \(\Gamma(U|X)\) can be decomposed into an encoding distribution \(Q(Z|X)\) and the generating distribution \(P_C(U|Z)\), and Theorem 1 mainly shows how to factor it through \(Z\).

In the first part, we will show that if \(P_C(U|Z)\) are Dirac measures, we have

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} = \inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\}, \tag{3}
\]

where \(\mathcal{P}(X \sim P_X, U \sim P_U)\) denotes the set of all joint distributions of \((X, U)\) with marginals \(P_X, P_U\), and likewise for \(\mathcal{P}(X \sim P_X, Z \sim P_Z)\). The set of all joint distributions of \((X, U, Z)\) such that \(X \sim P_X, (U, Z) \sim P_{C,z}\), and \((U \perp X)|Z\) are denoted by \(\mathcal{P}_{X,U,Z}\). \(P_{X,U}\) and \(P_{X,Z}\) denote the sets of marginals on \((X, U)\) and \((X, Z)\) induced by \(P_{X,U,Z}\).

From the definition, it is clear that \(\mathcal{P}_{X,U} \subseteq \mathcal{P}(P_X, P_C)\), therefore, we have

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} \leq \inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\}, \tag{4}
\]

The identity is satisfied if \(P_C(U|Z)\) are Dirac measures, such as \(U = C(Z)\). This is proved by the following Lemma in [50].

**Lemma 1** \(\mathcal{P}_{X,U} \subseteq \mathcal{P}(P_X, P_C)\) with identity if \(P_C(U|Z = z)\) are Dirac for all \(z \in Z\). (see details in [50].)

In the following part, we show that

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} = \inf_{Q_{XZ} \sim P_{CZ}} \mathbb{E}_{P_X \mathbb{E}_{Q(Z|X)}} \{\ell(f(X), C(Z))\}. \tag{5}
\]

Based on the definition, \(\mathcal{P}(P_X, P_C), \mathcal{P}_{X,U,Z}\) and \(\mathcal{P}_{X,U}\) depend on the choice of conditional distributions \(P_C(U|Z)\), but \(\mathcal{P}_{X,Z}\) does not. It is also easy to check that \(\mathcal{P}_{X,Z} = \mathcal{P}(X \sim P_X, Z \sim P_Z)\). The tower rule of expectation, and the conditional independence property of \(\mathcal{P}_{X,U,Z}\) implies

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} = \inf_{\Gamma \in \mathcal{P}(Y \sim P_Y)} \mathbb{E}_{(Y,U) \sim \Gamma} \{\ell(Y, U)\} \tag{6}
\]

Finally, since \(Y = f(X)\), it is easy to get

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} \tag{7}
\]

Now (3), (5) and (7) are proved and the three together prove Theorem 1.

Our proposed framework readily applies to non-deterministic case. If the classifier part is non-deterministic, Lemma 1 provides only the inclusion of sets \(\mathcal{P}_{X,U} \subseteq \mathcal{P}(P_X, P_C)\), and we can get an upper bound on the Wasserstein distance between the ground-truth and predicted label distributions:

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} \leq \inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\ell(f(X), U)\} \tag{8}
\]

where we assume the conditional distributions \(P_C(U|Z = z)\) have mean values \(C(z) \in \mathbb{R}^d\) and marginal variances \(\sigma_1^2, ..., \sigma_d^2 \geq 0\) for all \(z \in Z\), where \(C : Z \rightarrow \mathcal{X}\), and \(\ell(y, u) = \|y - u\|^2\). The above upper bound is derived by:

\[
\inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{(X,U) \sim \Gamma} \{\|f(X) - U\|^2\} = \inf_{\Gamma \in \mathcal{P}(X \sim P_X, U \sim P_U)} \mathbb{E}_{P_X \mathbb{E}_{P_U} \mathbb{E}_{P_Z}} \{\|f(X) - U\|^2\} \tag{9}
\]
and
\[
\mathbb{E}_{U \sim P(U|z)} \{ ||f(X) - U||^2 \} = \mathbb{E}_{U \sim P(U|z)} \{ ||f(X) - C(Z) + C(Z) - U||^2 \} = ||f(X) - C(Z)||^2 + \mathbb{E}_{U \sim P(U|z)} \{ ||C(Z) - U||^2 \} + \mathbb{E}_{U \sim P(U|z)} \{ < f(X) - C(Z), C(Z) - U > \}
\]
\[
= ||f(X) - C(Z)||^2 + \sum_{i=1}^{d} \sigma_i^2. \tag{10}
\]
In equation (10), the second term of the second last row becomes 0 since the optimization will drive \( f(X) - C(Z) \) to zero.

B. Implementation Details

B.1. Detection Model Training Algorithm

See details of training ER-Detector in Algorithm 2.

Algorithm 2: Training the Detection System

1: Input: Pre-trained encoder \( Q_\phi \).
2: \textbf{Training Procedure:}
3: Feeding all the training set images \( \{x_i\}_{i=1}^{n_{\text{train}}} \) into the encoder \( Q_\phi \) and get \( \{z_i\}_{i=1}^{n_{\text{train}}} \).
4: for \( t = 1, ..., m \) do \quad \triangleright \ m \text{ is the number of classes}
5: Fit KDE\(_t\) based on \( Z_i \), where \( Z_i \) is the set of \( z_i \) with label \( t \).
6: Generate noisy \( \{x_i^b\}_{i=1}^{\text{n\text{set}I}} \) and adversarial \( \{x_i^a\}_{i=1}^{\text{n\text{set}I}} \) examples based on \( \{x_i\}_{i=1}^{\text{n\text{train}}} \).
7: for \( i = 1, ..., \text{n\text{set}I} \) do
8: \( d_i = \text{KDE}\(_t\)(x_i) \), where the predicted label \( \hat{y}_i = t \)
9: \( d_i^b = \text{KDE}\(_t\)(x_i^b) \), where the predicted label \( \hat{y}_i^b = t \)
10: \( d_i^a = \text{KDE}\(_t\)(x_i^a) \), where the predicted label \( \hat{y}_i^a = t \)
11: Train Logistic Regression based on \( \{d_i\}_{i=1}^{\text{n\text{set}I}} \) and \( \{d_i^b\}_{i=1}^{\text{n\text{set}I}} \) as negative examples and \( \{d_i^a\}_{i=1}^{\text{n\text{set}I}} \) as positive examples.
12: \textbf{End Procedure}
13: \textbf{Return} \( G_\beta \)

B.2. Kernel Density Estimation

Kernel density estimation (KDE) is used in the detector \( G_\beta \) to model the low-dimensional space of the projection system. KDE is an unsupervised technique to estimate unknown probability distribution. Suppose that \( z_1, ..., z_n \) are training samples drawn from an unknown probability density \( f_Z(z) \). Given \( z \), we can use the following function to estimate the density score at \( z \):

\[
\hat{f}_Z(z) = \frac{1}{n} \sum_{i=1}^{n} K_\sigma(z, z_i),
\]

where \( K_\sigma(\cdot, \cdot) \) stands for kernel functions. In the experiments, one kernel density model is fitted for each class. Therefore, if \( z \) is predicted with label \( t \), the samples \( \{z_i\}_{i=1}^{n_{\text{train}}} \) used to do the estimation are training samples from class \( t \).

In the experiments, we apply the Gaussian kernel with bandwidth \( \sigma \):

\[
K_\sigma(z_1, z_2) \sim \exp(-\|z_1 - z_2\|^2/\sigma^2).
\]

The bandwidth parameter affects the “smoothness” of the resulting density. A large bandwidth leads to a very “smooth” density distribution. A small bandwidth usually leads to a “spiky” density distribution.

C. Experimental Details

C.1. Datasets

In this paper, we compare the performance of our proposed algorithm with other state-of-the-art defense methods on several benchmark datasets:

- MNIST [31]: handwritten digit dataset, which consists of 60,000 training images and 10,000 testing images. These are \( 28 \times 28 \) black and white images in ten different classes.
- CIFAR10 [29]: natural image dataset, which contains 50,000 training images and 10,000 testing images in ten different classes. These are low resolution \( 32 \times 32 \) color images.
- STL10 [11]: color image dataset similar to CIFAR10, but contains only 5,000 training images and 8,000 testing images in ten different classes. The images are of higher resolution \( 96 \times 96 \).
- Tiny Imagenet [14]: a subset of Imagenet dataset. Tiny Imagenet has 200 classes, and each class has 500 training images, 50 testing images, making it a challenging benchmark for the defense task. The resolution of the images is \( 64 \times 64 \).

Adversarial training parameters on different datasets are shown in Table 9. The parameters are the same for both Madry’s adversarial training and ER-Classifier for fair comparison.

<table>
<thead>
<tr>
<th>Data</th>
<th>( \epsilon )</th>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>0.3</td>
<td>40</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>0.03</td>
<td>20</td>
</tr>
<tr>
<td>STL10</td>
<td>0.03</td>
<td>20</td>
</tr>
<tr>
<td>Tiny Imagenet</td>
<td>0.01</td>
<td>10</td>
</tr>
</tbody>
</table>

C.2. Dimension of Embedding Space

One important hyper-parameter of ER-Classifier is the dimension of the embedding space. If the dimension is too small, important features are “collapsed” onto the same dimension, and if the dimension is too large, it will be hard
to regularize the embedding and result in too much noise and instability. The maximum likelihood estimation of intrinsic dimension proposed by [33]\(^3\) is used to calculate the intrinsic dimension of each image dataset, serving as a guide for selecting the embedding dimension. The sample size used in calculating the intrinsic dimension is 1,000, and increasing the sample size does not influence the results much. Based on the intrinsic dimension estimated by [33], we test several different values around the estimated intrinsic dimension and evaluate the models against the \(l_\infty\)-PGD attack. All models are trained without min-max robust optimization, and the experimental results are shown in Figure 7.

The final embedding dimension is chosen based on robustness, number of parameters, and testing accuracy when there is no attack. The final embedding dimensions and estimated intrinsic dimensions are shown in Table 10.

Table 10. Pixel space dimension, intrinsic dimension calculated by [33], and final embedding dimension used.

<table>
<thead>
<tr>
<th>Data</th>
<th>Data dim.</th>
<th>Estimated Intrinsic dim.</th>
<th>Embedding dim.</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST</td>
<td>1 × 28 × 28</td>
<td>13</td>
<td>4</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>3 × 32 × 32</td>
<td>17</td>
<td>16</td>
</tr>
<tr>
<td>STL10</td>
<td>3 × 96 × 96</td>
<td>20</td>
<td>16</td>
</tr>
<tr>
<td>Tiny Imagenet</td>
<td>3 × 64 × 64</td>
<td>19</td>
<td>20</td>
</tr>
</tbody>
</table>

Based on Figure 7, the embedding dimension close to the estimated intrinsic dimension usually offers better results except on MNIST. One explanation may be that MNIST is a simple handwritten digit dataset, so performing classification on MNIST may not require that many dimensions.

C.3. Epsilon Selection

Epsilon (\(\epsilon\)) is an important hyper-parameter for adversarial training. When doing Madry’s adversarial training, we test the model robustness with different \(\epsilon\) and choose the best one. The experiment results are shown in Figure 8.

Based on Figure 8, we use \(\epsilon = 0.3, 0.03, 0.03\) in Madry’s adversarial training on MNIST, CIFAR10 and STL10 respectively. For Tiny Imagenet, we use \(\epsilon = 0.01\). To make a fair comparison, we use the same \(\epsilon\) when training ER-Classifier.

C.4. Prior Selection

ER-Classifier does not have restrictions on the choice of prior. However, it is interesting to explore the performances of different priors.

Three different prior distributions are tested on MNIST and CIFAR10 datasets. They are standard Gaussian, Uniform(\([-3, 3]\)) and Cauchy(0, 1), where Cauchy(0, 1) has the same support as standard Gaussian but is heavy tailed and 99.7% of the standard Gaussian points lies within \([-3, 3]\). All the models are trained without min-max robust optimization, and the experimental results are shown in Figure 9. Based on the results, all three priors work well, but standard Gaussian performs best on both datasets.

Ding et al. [17] prove that adversarial robustness is sensitive to the input data distribution, and if the data is uniformly distributed in the input space, no algorithm can achieve good robustness. They also empirically show that cornered/concentrated data distributions tend to achieve better robustness. Standard Gaussian pushes the embedding space to be more concentrated, making the valid perturbation space to be smaller. This may explain why Gaussian prior performs a little bit better than two other priors.

D. More Detection Experiments

D.1. Testing Against High Confidence Adversarial Examples

In [2], the author pointed out that LID detection method is not able to detect high confidence adversarial examples generated by C&W attack. This might be a concern for all detection methods. Therefore, in this part, we apply C&W to generate high confidence adversarial examples and test the detectors against them.

We generate 100 high confidence adversarial examples for both datasets and evaluate the detectors against them. The confidences are 9 and 20 for MNIST and CIFAR10 respectively. Based on the experiment, if the confidence goes higher than those thresholds, it is difficult to generate adversarial examples on the corresponding datasets within the \(L_2\) thresholds. The performances of the detectors are shown in Tables 11-13.

Table 11. Trained on FGSM adv.examples and tested on high confidence adv.examples generated by C&W.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST (ACC-NADV, ACC-COM)</th>
<th>CIFAR10 (ACC-NADV, ACC-COM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>63.50, 11.00</td>
<td>86.50, 16.00</td>
</tr>
<tr>
<td>LID</td>
<td>57.00, 11.00</td>
<td>43.00, 16.00</td>
</tr>
<tr>
<td>ER1</td>
<td>80.00, 100.00</td>
<td>82.50, 100.00</td>
</tr>
<tr>
<td>ER2</td>
<td>83.00, 100.00</td>
<td>77.00, 16.00</td>
</tr>
</tbody>
</table>

Table 12. Trained on PGD adv.examples and tested on high confidence adv.examples generated by C&W.

<table>
<thead>
<tr>
<th>Methods</th>
<th>MNIST (ACC-NADV, ACC-COM)</th>
<th>CIFAR10 (ACC-NADV, ACC-COM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>KD</td>
<td>74.00, 11.00</td>
<td>74.50, 16.00</td>
</tr>
<tr>
<td>LID</td>
<td>83.00, 11.00</td>
<td>94.00, 36.00</td>
</tr>
<tr>
<td>ER1</td>
<td>73.00, 100.00</td>
<td>79.00, 100.00</td>
</tr>
<tr>
<td>ER2</td>
<td>80.50, 100.00</td>
<td>72.50, 16.00</td>
</tr>
</tbody>
</table>

\(^3\)Code publicly available at [https://github.com/OFAI/hub-toolbox-python3](https://github.com/OFAI/hub-toolbox-python3)
D.2. Effect of Regularization

To show that regularization on the embedding space helps improve the robustness of ER-Detector, we fit frameworks with only the encoder and classifier part (ER1− and ER2−), where the encoder and classifier have the same structures as in ER-Detector. The results are shown in Table 14. Considering both ACC-NADV and ACC-COM, ER-Detector performs much better than structures without regularization. Instead of fitting a discriminator to regularize the embedding space, Kullback-Leibler distance is also tried. However, the KL loss and classification loss cannot converge together during the training.

E. Loss Surface Visualization

To show that ER-Classifier outperforms Madry’s adversarial training is not because of weird loss surface, we visualized the loss surfaces of ER-Classifier and Madry’s adversarial training on CIFAR10. Following the implementation in [18], we vary the data input along a linear space defined by the sign of the input gradient and a random Rademacher vector, where the x- and y-axes represent the magnitude of the perturbation added in each direction and the z-axis represents the loss. Based on the results in Figure 10. We can see that both methods have smooth loss surfaces.

F. Embedding Visualization

Larger versions of embedding visualization plots are shown in Figure 11 and Figure 12.
Figure 9. Testing accuracy of models with different prior distributions under $l_\infty$-PGD attack.

Table 14. Performances of ERs$^-$ and ERs on MNIST and CIFAR10 against PGD, FGSM and C&W attacks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>PGD</th>
<th>FGSM</th>
<th>C&amp;W</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ER1</td>
<td>ER1</td>
<td>ER2</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>ACC-NADV 97.68</td>
<td>ACC-COM 58.74</td>
<td>ACC-NADV 97.54</td>
</tr>
</tbody>
</table>

G. Model Structure

MNIST, CIFAR10, STL10 and TinyImagenet classifier structures used for baseline methods are shown in Table 15. Details of ER-Classifier structures on the four benchmark datasets are shown in Table 16. The discriminator architecture is the same on four datasets: four fully connected layers. See code of model in code files in supplementary code folder.

Table 15. Architectures of baseline networks.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST [31]</td>
<td>4Conv. + 4FC layers</td>
</tr>
<tr>
<td>CIFAR10 [29]</td>
<td>VGG19 with BN [47]</td>
</tr>
<tr>
<td>STL10 [11]</td>
<td>6Conv. with BN and 5Max.Pool +4FC</td>
</tr>
<tr>
<td>Tiny Imagenet [15]</td>
<td>13Conv. with BN and 5Max.Pool +4FC</td>
</tr>
</tbody>
</table>

Table 16. Architectures of ER-Classifier Encoders.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Encoder Architecture</th>
<th>Classifier Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>MNIST [31]</td>
<td>4Conv. with BN +1FC</td>
<td>3FC with BN</td>
</tr>
<tr>
<td>CIFAR10 [29]</td>
<td>16Conv. + 1FC</td>
<td>4FC</td>
</tr>
<tr>
<td>STL10 [11]</td>
<td>6Conv. with BN +1FC</td>
<td>3FC</td>
</tr>
<tr>
<td>Tiny Imagenet [15]</td>
<td>13Conv. with BN and 5Max.Pool +1FC</td>
<td>3FC</td>
</tr>
</tbody>
</table>
Figure 10. Loss surfaces of ER-Classifier and Madry’s adversarial training. (Left: ER-Classifier, Right: Madry’s adversarial training)

Figure 11. 2D embeddings for E-CLA and ER-Classifier on MNIST.
Figure 12. 2D embeddings for E-CLA and ER-Classifier on CIFAR10.