Task-Agnostic Robust Representation Learning

A. Tuan Nguyen[*] [†] Ser Nam Lim[‡] Philip Torr[*]

March 9, 2022

Abstract

It has been reported that deep learning models are extremely vulnerable to small but intentionally chosen perturbations of its input. In particular, a deep network, despite its near-optimal accuracy on the clean images, often mis-classifies an image with a worst-case but humanly imperceptible perturbation (so-called adversarial examples). To tackle this problem, a great amount of research has been done to study the training procedure of a network to improve its robustness. However, most of the research so far has focused on the case of supervised learning. With the increasing popularity of self-supervised learning methods, it is also important to study and improve the robustness of their resulting representation on the downstream tasks. In this paper, we study the problem of robust representation learning with unlabeled data in a task-agnostic manner. Specifically, we first derive an upper bound on the adversarial loss of a prediction model (which is based on the learned representation) on any downstream task, using its loss on the clean data and a robustness regularizer. Moreover, the regularizer is task-independent, thus we propose to minimize it directly during the representation learning phase to make the downstream prediction model more robust. Extensive experiments show that our method achieves preferable adversarial performance compared to relevant baselines.

1 Introduction

Deep learning has achieved state-of-the-art performance in many tasks such as image classification, object detection, and natural language processing. Instead of having to select handcrafted features and representation of the input as in classical machine learning, deep learning has the ability to automatically learn a meaningful representation with deep networks and gradient descent. However, the success of deep learning relies on the availability of a large amount of labeled data, which is expensive in practice. Self-supervised learning [1, 2, 3, 4] has gained interest as a solution to the above problem due to its ability to learn from unlabeled data. However, although the representation learned via self-supervised learning is often meaningful for downstream tasks, the resulting prediction model usually lacks adversarial robustness. Currently, to the best of our knowledge, virtually all existing adversarial training methods [5, 6] require labels and/or the prediction task. Furthermore, for computational reasons, we do not want to re-do the expensive adversarial training for the downstream tasks (this is also in the spirit of self-supervised learning); therefore, the (theoretical) transferability of a robust representation among different tasks is also crucial. However, this transferability aspect of robustness has not been well-studied. For example, [7] proposes a framework for adversarial contrastive learning, which enforces the robustness of the representation network by finding worst-case adversaries that maximize the contrastive loss, followed by minimizing that loss with respect to the network parameters. However, it is not clear how this robustness can be transferred to a downstream task.

In this paper, we develop a task-independent robust representation learning method to tackle the above issue. Intuitively, if we could enforce the representation z of an input x to be close to that of its neighbors in the adversarial ball (for example, a l_∞ ball around x), then it would make the downstream model more robust (Figure 1). In particular, any adversarial example x^{adv} of x produced by an attack on the downstream task is within this adversarial ball, thus its representation would be close to that of x, making it harder for the task's decision boundary to separate them. However, there are some challenges when applying this idea naively. One might think that we can find a worst-case adversarial example $x^{u\text{-}adv}$ (unsupervised adversary, as opposed to the typical supervised adversary x^{adv}) with the largest distance to x in the representation space (based on some distance metrics), and minimize that distance with respect to the network's parameters. This, however, is problematic for a typical deterministic representation network, since it is not trivial how to define the distance on the representation space (for example, using the l_2 distance between the representations would not help since the model can "cheat" by making the norm of the representation smaller, which can be easily compensated by making the weights of the next fully connected layer bigger). Fortunately, in this paper, we observe that with a probabilistic representation network $p_\theta(z|x)$, we can define the "closeness" of the representations by the KL

[*]University of Oxford
[†]Corresponding author: tuan@robots.ox.ac.uk
[‡]Meta AI Research

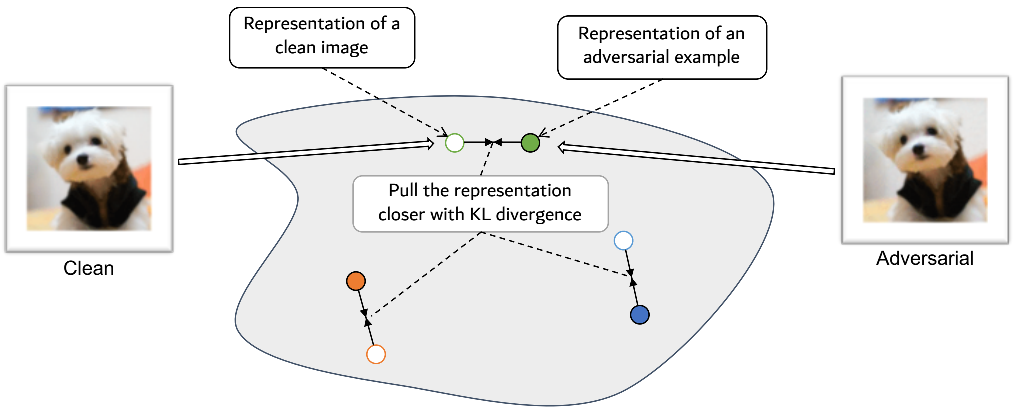
Representation of a clean image
Representation of an adversarial example
Pull the representation closer with KL divergence
Clean
Adversarial

Figure 1: **Illustration of our proposed regularizer**. This figure illustrates the (probabilistic) representation space, with each circle representing (the distribution of) the representation of an image. Non-filled circles depict the (probabilistic) representations of natural images and filled circles depict that of their adversarial examples. We use the KL divergence to pull the representations of an image and its adversarial images closer, which improves the bound of the adversarial loss on a downstream task.

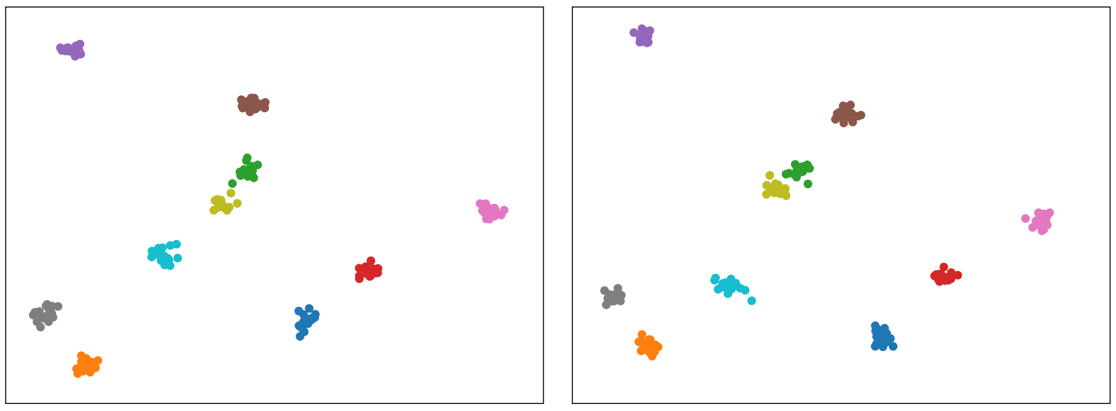

Figure 2: **Visualization of the probablistic representation of our method**. Each color corresponds to a single image. For each image x, we sample 20 z's from the probabilistic representation distribution $p_\theta(z|x)$ (hence the clusters of points). The left figure is for the original images and the right figure is for the unsupervised adversaries. Our method enforces the representation distribution of an image to be close to that of its adversaries, making the downstream models more robust.

divergence, which allows for creating adversarial examples during unsupervised/self-supervised learning via an inner maximization of the KL term ($\mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\text{-}adv})]$). Visualization of the representation distribution of an image and its adversarial example of our method can be found in Figure 2. We also show that this leads to a robustness regularizer for unsupervised learning that is **provably transferable** to downstream tasks (more details in the Approach section). Note that this idea can also be thought of as enforcing a low Lipschitzness of the representation network. However, as mentioned above, a typical norm-based (e.g., l_1 and l_2) Lipschitzness in the representation space is not meaningful (due to the aforementioned "cheat"), whereas using KL divergence solves the problem and also leads to a theoretical guarantee on the downstream robustness.

In this work, we first propose an upper bound of the adversarial loss of a model for a certain prediction task based on its loss on the clean data and a robustness regularizer. The regularizer, which is based on the KL divergence as described above, only depends on the representation mapping and is independent of the task. Therefore, we can minimize this regularizer term during the representation learning phase (of an unsupervised or self-supervised method), so that the robustness will be transferred directly to a prediction model when the representation is used for a downstream task. Our method can be straightforwardly applied to any representation learning method, including unsupervised learning (e.g., VAE) or any self-supervised learning model.

Our contributions in this work are threefold:

- We derive an upper bound on the adversarial loss of a prediction model on a certain task, based on a task-independent robustness regularizer.
- We propose to incorporate the above regularizer into existing representation learning frameworks to improve the adversarial robustness of the representation on downstream tasks.
- We demonstrate the use of our proposed regularizer with existing representation learning frameworks, and show that the resulting models achieve SOTA results in the unsupervised robust representation learning task.

2 Related Works

2.1 Adversarial Training

It has been shown that neural networks, even with a high classification accuracy, are vulnerable to small bounded (but intentionally worst-case chosen) adversarial perturbations [8]. From this observation, many methods have been proposed to alter the typical training procedure of a neural network to improve its adversarial robustness (hence the name adversarial training). Two of the most often-used methods are AT [5] and TRADES [6], in which the algorithms try to find a worst-case perturbation of the input (called an adversarial example) with an inner maximization problem and minimize the loss (or a regularizer) with respect to that adversarial example. Studying the inner maximization problem is also an active and attractive research direction, with the aim to develop fast and/or accurate methods to find the adversarial examples. For example, PGD [5] uses multiple steps of projected gradient descent, which is accurate (and somewhat "gold-standard") but expensive. For this computational reason, many one-step algorithms have been proposed, including FGSM [9], RS-FGSM [10] and GradAlign [11], with varying level of success. These adversarial defense/attack methods are relevant to our work since we also need to solve an inner maximization problem to find (unsupervised) adversarial examples for the robustness regularizer.

2.2 Self-Supervised Learning

As mentioned earlier, self-supervised learning has received great interest due to its ability to learn from unlabeled data, which reduces the need for expensive annotations of images. Self-supervised learning is based on a user-defined pretext task, which can be as simple as to predict the rotation angle of an image or more complex such as to solve a Jigsaw puzzle [12]. Recently, a popular and successful self-supervised learning paradigm is to learn representations that are invariant under different augmentations (also referred to as 'distortions') of an image [1, 4, 3]. The idea of this learning paradigm is to maximize the similarity between representations of two augmentations of an image, while avoiding network collapse (to a trivial and meaningless solution such as a constant function) by different objective functions. Since our proposed robustness regularizer is task-independent, it can be straightforwardly applied to most of these self-supervised learning methods.

2.3 Robust Self-Supervised Learning

Recently, the field of unsupervised robust representation learning has gained increasing interest, with the goal to leverage unlabeled data and learn a robust and meaningful representation for downstream tasks. One of the main baselines to our work is RoCL [7], which applies adversarial training directly on the contrastive loss in SimCLR. However, it is not clear if a representation that is robust to the contrastive loss will be robust to the prediction loss of a downstream task. Some other methods [13, 14] perform adversarial training on the self-supervised task to aid the adversarial training of the main task (not to replace), which are less related to our work. Similarly, [15] utilizes unlabeled data to help train a robust classifier in a semi-supervised manner. Meanwhile, [16] propose an algorithm to improve the robustness of PCA. However, PCA is not a common component of modern deep learning architectures/pipelines, thus its practicality might be limited.

3 Approach

3.1 Problem statement

Assume that we have a distribution $p(x)$ of data (e.g., images), with $x \in \mathcal{X}$ as the input. We are interested in the problem of robust representation learning, where we want to learn a representation z of x with the mapping $p_\theta(z|x)$, parameterized by θ; so that z is meaningful for downstream tasks (to be defined below), and that any classifier (based on z) of the downstream tasks should be robust against adversarial attacks. For computational reasons, we do not want to re-do the adversarial training, and desire that the robustness transfer directly to the downstream tasks. The representation z can be probabilistic (e.g., $p_\theta(z|x) = \mathcal{N}(z; \mu_\theta(x), \sigma_\theta(x))$) or deterministic (i.e., $p_\theta(z|x) = \delta_{g_\theta(x)}(z)$ with a deterministic function g_θ). In this paper, we will especially consider a probabilistic representation mapping. In practice, the representation can be learn by unsupervised learning methods (e.g., VAE) or self-supervised learning methods (with a pretext task).

Remark 1. *A note on the choice of the representation distribution $p_\theta(z|x)$.*

As mentioned earlier, in this paper, we especially consider a probabilistic representation mapping. Specifically, we use a Gaussian distribution in all of our experiments, e.g., $p_\theta(z|x) = \mathcal{N}(z; \mu_\theta(x), \sigma_\theta(x))$. This is just a design choice that is simple and works well in practice, and our work is not limited by this choice of the distribution. We can use almost any other distribution (with a known parameterization trick to allow for backpropagation). Also, note that the Gaussian representation network is a generalized version of a deterministic network (it becomes a deterministic network when $\sigma_\theta(x) \to 0 \; \forall x$); therefore, this network choice is not at all restricted when compared to a typical deterministic network.

Any downstream task T is defined by a conditional distribution $p_T(y|x)$ where $y \in \mathcal{Y}$ is the label. The joint data distribution of this task is $p_T(x,y) = p(x)p_T(y|x)$. With the representation mapping $p_\theta(z|x)$ learned in advance, we want to learn a classifier $\hat{p}_T(y|z)$ (parameterized by ω_T, which we will omit for notation simplicity) for the task T. This is often called an output head that classifies y given z.
The predictive distribution of y given x for this task T is:

$$\mathbb{E}_{p_\theta(z|x)}[\hat{p}_T(y|z)] \tag{1}$$

(for a deterministic representation mapping, Eq. 1 simplifies into $\hat{p}_T(y|z = g_\theta(x))$)

Remark 2. *On the inference complexity of a probabilistic representation.*

Using a probabilistic representation, we need to sample multiple z from $p_\theta(z|x)$ to estimate Eq. 1 with Monte Carlo sampling during test time. However, this is not a big issue for the representation learning framework, since we only need to run the representation network $p_\theta(z|x)$ (which is usually deep) once to get a distribution of z. After sampling multiple z from that distribution, we only need to rerun the classifier $\hat{p}_T(y|z)$, which is usually a small network (e.g., often contains one or a few fully-connected layers). Furthermore, we can also run $\hat{p}_T(y|z)$ (a small network) in parallel for multiple z to reduce inference time if necessary.

During training of the downstream task T, a single z is sampled per input x from the learned representation mapping $p_\theta(z|x)$; and the output head $\hat{p}_T(y|z)$ is trained via minimizing the following training objective:

$$\mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right] \tag{2}$$

With common choices of the predictive distribution $\hat{p}_T(y|z)$, the quantity $-\log \hat{p}_T(y|z)$ is often non-negative. For example, with a categorical predictive distribution in a classification problem, this term is the cross-entropy loss; whereas with a Gaussian predictive distribution (with a fixed variance) in a regression problem, the above term becomes the squared error loss (with an additive constant).

Note also that the objective in Eq. 2 is an upper bound of the true loss for task T $\mathbb{E}_{p_T(x,y)}\left[-\log \mathbb{E}_{p_\theta(z|x)}[\hat{p}_T(y|z)]\right]$ (due to Jensen's inequality), where $-\log \mathbb{E}_{p_\theta(z|x)}[\hat{p}_T(y|z)]$ is the loss of a datapoint (x,y).

Now we formally define the adversarial robustness of the network on the downstream task. Denote $A(x)$ to be the set of adversarial examples of x. This set is different for different kinds of adversarial attack; for example, with an l_∞ attack, $A(x)$ is the l_∞-ball around x with a predefined radius of ϵ. The adversarial loss of the task T is:

$$\mathbb{E}_{p_T(x,y)}\left[\max_{x^{adv}\in A(x)} -\log \mathbb{E}_{p_\theta(z|x^{adv})}[\hat{p}_T(y|z)]\right] \tag{3}$$

Intuitively, this means that an attacker seeks to find an adversarial example x^{adv} of each input x that maximizes the loss w.r.t. its label y; and we, as the defender, want to minimize that loss. In the next subsections, we will discuss how we can minimize this adversarial loss, even in a task-agnostic manner during the representation learning phase.

3.2 A bound on the adversarial loss

We first propose a bound on the adversarial loss based on the downstream training objective and a robustness regularizer:

Proposition 1. *Assuming that $\forall x$, $p_\theta(z|x)$ has the same support set $\mathcal{Z}$ (e.g., $p_\theta(z|x)$ is Gaussian); and that $-\log \hat{p}_T(y|z) \leq M\ \forall z \in \mathcal{Z}, y \in \mathcal{Y}$ [1], we have:*

$$\begin{aligned}\mathbb{E}_{p_T(x,y)}\left[\max_{x^{adv}\in A(x)} -\log \mathbb{E}_{p_\theta(z|x^{adv})}[\hat{p}_T(y|z)]\right] \leq\ & \mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right] \\ & + \frac{M}{\sqrt{2}}\sqrt{\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv}\in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]}\end{aligned} \tag{4}$$

Proof. provided in the supplementary file. □

The first term $\mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right]$ is the downstream training loss in Eq. 2, and will be minimized during the training of the output head $\hat{p}_T(y|z)$ for task T.

We call the second term $\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv}\in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]$ a robustness regularizer. Since we do not want to perform adversarial training for the downstream tasks, we want to minimize this term during the representation learning phase. Since this term is label-free and task-independent, if we minimize it during the representation learning phase, it will transfer directly to the downstream task and help minimize the bound in Eq. 4. This will be discussed further

[1] In the classification problem, we can enforce this quite easily by augmenting the output softmax of the classifier $\hat{p}_T(y|z)$ so that each class probability is always at least $\exp(-M)$. For example, if we choose $M = 3 \Rightarrow \exp(-M) \approx 0.05$, and if the output softmax is $(p_1, p_2, ..., p_C)$, we can augment it into $(p_1 \cdot K + 0.05, p_2 \cdot K + 0.05, ..., p_C \cdot K + 0.05)$, where $K = 1 - 0.05 \cdot C$ and C is the number of classes. This ensures the bound for the loss of a datapoint, while retaining the output prediction class.

in Subsection 3.3. Recall that we use the Gaussian distribution of the per-image representation network, i.e., $p_\theta(z|x) = \mathcal{N}(z; \mu_\theta(x), \sigma_\theta(x))$, so this KL term can be computed analytically (and exactly). Also note that, as discussed in Remark 1, the Gaussian representation is a generalized version of a typical deterministic representation, so it is sufficiently expressive for typical Deep Learning problems.

Comparison between our bound and TRADES [6]:

- The bound in TRADES only works for the case of binary classification, while our bound works for the general case of supervised learning (including multi-class classification and regression).
- Our robustness regularizer is label-free and task-independent. Therefore, we can minimize it in the representation learning phase (with unsupervised or self-supervised tasks), and it will transfer directly to the downstream tasks. On the other hand, the robustness regularizer in TRADES is task-dependent, thus minimizing the term for a pretext task does not necessarily transfer to the downstream task. Furthermore, TRADES's robustness regularizer requires a predictive distribution of a task to compute, and this might not be applicable to many self-supervised learning methods where there is no prediction task (e.g., contrastive learning). These arguments are also true for almost all existing robustness methods.

Trade-off between Clean Accuracy and Adversarial Robustness The trade-off between a model's performance on clean input and adversarial input has been well observed in practice [6]; and this phenomenon can also be explained with our bound. Minimizing the first term in Eq. 4 will help the model's performance on clean input, while minimizing the second term increase the model's robustness against adversarial input; and there is an inherent trade-off between them. Minimizing the second term $\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]$ too much will compress the representation, hurting its expressiveness and separability among classes. For example, consider the l_2 defense with radius ϵ ($A(x)$ will be the l_2-ball around x with radius ϵ). The above regularizer encourages the representation distribution of an input x to be similar to that of its neighbours in the l_2-ball. Now, if there exist two inputs x_1 and x_2 from different classes such that $\epsilon < ||x_1 - x_2||_2 < 2\epsilon$, then these two points do not belong to the other's adversarial set. Let $x' = (x_1 + x_2)/2$, it follows that $||x_1 - x'||_2 = ||x_2 - x'||_2 = ||x_1 - x_2||_2/2 < \epsilon$, meaning $x' \in A(x_1)$ and $x' \in A(x_2)$. Note that minimizing the regularizer term too much will encourage the representation distribution of both x_1 and x_2 to be similar to that of x'; and the classifier might fail to separate the two datapoints. Therefore, it might hurt the expressiveness and separability of the representation, especially for the inputs around the decision boundary.

3.3 Applications and Use Cases

In this subsection, we will discuss the use of our robustness regularizer $\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]$. Although this term can be used directly in a supervised learning setting, a far more exiting application is to minimize it during the representation learning phase (of an unsupervised or self-supervised method). As mentioned earlier, since this term is task-independent, if we minimize it during the representation learning phase, it will transfer directly to downtream tasks, improving the model's adversarial robustness on these tasks (see Eq. 4). We name our regularizer **Urkle** (**U**nsupervised **R**obustness with **KL** divergenc**E**). In this subsection, we demonstrate some example scenarios to learn a meaningful (and robust) representation, namely with unsupervised learning (via VAE) and self-supervised learning (with any pretext task).

3.3.1 With VAE

In VAE [17], we have an encoder $p_\theta(z|x)$ (which also acts as our representation mapping), a decoder $q_\phi(x|z)$, and a prior $p(z)$, the objective of VAE (negative ELBO) is:

$$\mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log q_\phi(x|z)]\right] + \mathbb{E}_{p(x)}\left[\mathrm{KL}[p_\theta(z|x)|p(z)]\right] \tag{5}$$

Here we add our robustness regularizer to learn a robust encoder $p_\theta(z|x)$, leading to the below objective:

$$\begin{aligned}\mathbb{E}_{p(x)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log q_\phi(x|z)]\right] + \beta_{VAE}\mathbb{E}_{p(x)}\left[\mathrm{KL}[p_\theta(z|x)|p(z)]\right] \\ +\beta_{robust}\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]\end{aligned} \tag{6}$$

Note that we also add a coefficient β_{VAE} for the VAE's regularizer $\mathbb{E}_{p(x)}\left[\mathrm{KL}[p_\theta(z|x)|p(z)]\right]$ (similar to β-VAE [18]). We use a small value of β_{VAE} in practice since we found that this term might hinder the expressiveness of the representation $p_\theta(z|x)$.

All three expectation terms in Eq 6 can be estimated with a minibatch of input x's. For each input x, we find the unsupervised adversary x^{u_adv} by the inner maximization problem of $\max_{x^{u\_adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]$ with, for example, the PGD algorithm. The objective in Eq 6 is minimized with respect to θ and ϕ (thus it will learn an encoder $p_\theta(z|x)$ such that the representation of an input is close to that of its adversaries).

3.3.2 With a self-supervised learning task

Let's assume we have a pretext task designed to learn a meaningful representation of x. Since self-supervised tasks and their loss functions are diverse, we will refer to the loss function as $L_{ssl}(p(x), p_\theta(z|x), \phi)$ in general, where $p_\theta(z|x)$ is a representation mapping used to solve the task (e.g., might be used for a pretext classification task, to solve a jigsaw puzzle, or to minimize the contrastive loss in the contrastive learning framework) and ϕ is any additional parameters (apart from θ) used for this self-supervised task (e.g., parameters of the projector in SimCLR [1], parameters of the output head of some pretext classification task).

Similarly, we can also add the robustness regularizer term here to to learn a robust representation mapping $p_\theta(z|x)$, leading to the following objective:

$$L_{ssl}(p(x), p_\theta(z|x), \phi) + \beta_{robust}\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv}\in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right] \tag{7}$$

Demonstration with SimCLR [1]: We re-emphasize that our proposed robustness regularizer can be applied to almost all unsupervised and self-supervised representation learning methods. However, since our main baseline [7] is built upon SimCLR, we also use SimCLR (with a slight adaptation of using a probabilistic representation network) in our experiments for a fair comparison (note that we conjecture using more recent and advanced self-supervised learning methods [4, 2] will likely improve further the performance of our model). We demonstrate the resulting method here in details.

First of all, SimCLR (and other contrastive learning methods) learns from two augmentations of each image. Let denote a minibatch of data as $\{(x_{1,1}, x_{1,2}), (x_{2,1}, x_{2,2}), ..., (x_{b,1}, x_{b,2})\}$, where $x_{i,1}$ and $x_{i,2}$ are two augmented version of a same original image x_i, and are called positive examples of each other. The goal of contrastive learning is to encourage the representation to be similar among the positive images. For each x we sample a single z from the network $p_\theta(z|x)$ to make the minibatch of representation $\{(z_{1,1}, z_{1,2}), (z_{2,1}, z_{2,2}), ..., (z_{b,1}, z_{b,2})\}$. Then the loss function of SimCLR is:

$$\ell_{SimCLR} = \ell_{NT\_Xent}((z_{1,1}, z_{1,2}), (z_{2,1}, z_{2,2}), ..., (z_{b,1}, z_{b,2})) \tag{8}$$

where ℓ_{NT_Xent} is the so-called "normalized temperature-scaled cross entropy" loss function that takes input as a batch of tuples $t_1, t_2, ..., t_b$, with each t_i is a tuple of representations from a set of positive images (in the case above each tuple would be of length 2). Specifically:

$$\ell_{NT\_Xent}(t_1, t_2, ..., t_b) = \sum_{i=1}^{b} \sum_{z,z'\in t_i, z\neq z'} \frac{\exp(\mathrm{sim}(z,z')/\tau)}{\sum_{j\neq i}\sum_{z''\in t_j}\exp(\mathrm{sim}(z,z'')/\tau)} \tag{9}$$

where $\mathrm{sim}(z, z')$ is the cosine similarity between z and z' (we also often project the representation z to a lower dimensional space before calculating the cosine similarity), and τ is the temperature (often set to 0.5).

To implement our regularizer, we find the unsupervised adversaries $\{(x^{u\_adv}_{1,1}, x^{u\_adv}_{1,2}), ..., (x^{u\_adv}_{b,1}, x^{u\_adv}_{b,2})\}$ of $\{(x_{1,1}, x_{1,2}), ..., (x_{b,1}, x_{b,2})\}$ that maximize:

$$\frac{1}{2b}\sum_{i=1}^{b}\sum_{k=1}^{2} \max_{x^{u\_adv}_{i,k}\in A(x_{i,k})} \mathrm{KL}[p_\theta(z|x_{i,k}))|p_\theta(z|x^{u\_adv}_{i,k}))] \tag{10}$$

Following [7], we also use PGD [5] to solve this inner maximization problem.

Let ℓ_{Urkle} be the value of the above maximum, i.e.:

$$\ell_{Urkle} = \frac{1}{2b}\sum_{i=1}^{b}\sum_{k=1}^{2} \mathrm{KL}[p_\theta(z|x_{i,k}))|p_\theta(z|x^{u\_adv}_{i,k}))] \tag{11}$$

with $\{x^{u\_adv}_{i,k}\}$ found above.

Now we can add the regularizer ℓ_{Urkle} directly to the original loss function ℓ_{SimCLR}. However, we note that $x^{u\_adv}_{i,1}$ and $x^{u\_adv}_{i,2}$ can also be treated as positive images of $x_{i,1}$ and $x_{i,2}$; therefore, we also include them when compute the *NT_Xent* loss. The final loss function of our model as:

$$\ell_{NT\_Xent}((z_{i,1}, z_{i,2}, z^{u\_adv}_{i,1}, z^{u\_adv}_{i,2})_{i=1}^{b}) + \beta_{robust}\ell_{Urkle} \tag{12}$$

where $z^{u\_adv}_{i,k} \sim p_\theta(z|x^{u\_adv}_{i,k})\ \forall i \in \overline{1,b}, k \in \{1,2\}$, ℓ_{NT_Xent} is calculated as in Eq 9 (in this case each tuple is of length 4), and β_{robust} is a hyperparameter.

PGD attack/defense: Since Projected Gradient Descent is used in both training and evaluation in our experiments, we briefly review it here to make the paper more self-contained. Let's assume that we need to find x' within an l_∞-norm (or other norms) ball of radius ϵ around x that maximizes the function $f(x')$. The PGD algorithm is as follows:

1. Initialize x^0 to x (possibly with a small added random noise).

2. Update $x^i = x^{i-1} + \alpha \texttt{sign}(\nabla_{x^{i-1}} f(x^{i-1}))$ with a step size α, and clip x^i to the ϵ-ball that is being considered.

3. Repeat step 2 k times, and set $x' = x^k$.

Note that we can also use any other adversarial attack/defense methods for the inner maximization problem. We use PGD in this paper because it is considered "gold-standard" at the moment, and it is also used by our main baseline [7].

4 Experiments

We conduct extensive experiments to validate our method. In this section, we describe these experiments in details. For more information regarding the experimental settings and the baselines, please refer to our supplementary file.

4.1 Datasets

MNIST [19] contains 70000 images of hand-written digits with the classification task of 10 digits.

CIFAR10 [20] consists of 60000 images of size 32x32, and over ten classes: airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck.

CIFAR100 [20] Similar to CIFAR10, CIFAR100 also consists of 60000 images with the size 32x32. The task is classification with 100 different classes.

ImageNet [21] is a large scale real-world computer vision dataset, which consists of 1000 classes. To the best of our knowledge, this dataset has not been considered by previous adversarial self-supervised learning methods.

4.2 Experimental Settings

Within each experiment, we use the same network as the representation network for all models. Since our representation network is probabilistic, it only differs from the other deterministic networks in the last layer. In particular, for a representation of size d_z, the last layer's dimension of a deterministic representation network is d_z, while that of a probabilistic network is $2 \cdot d_z$ (d_z for μ and d_z for σ^2). Also, although our method can be used for any adversarial attack (e.g., l_1 or l_2), we consider the l_∞ adversaries in this experiments section. This is because l_∞ is one of the most common adversarial attacks, and it is also used in our main baseline [7].

With VAE: We test the effectiveness of our robustness regularizer when used with VAE (as described in Section 3.3.1) with the MNIST dataset. The main baseline we consider in this experiment is AE (auto-encoder) with a TRADES-like regularizer. This is because the reconstruction in AE can be viewed as a prediction task, so we can use the regularizer in TRADES to force the reconstruction of an image to be similar to the reconstruction of its adversaries (more details of this baseline in the supplementary file). To make the comparison fair for AE (that has no built-in regularizer), we set $\beta_{VAE} = 0$ in our experiment, although we note that slightly increasing this value leads to even better representation. Apart from this baseline, we also include supervised learning models (Standard Training, AT [5] and TRADES [6]) for reference. For this experiment, we consider the l_∞ perturbation with $\epsilon = 0.1$. For this "toy" experiment, we use a simple convolutional neural network with four 3×3 convolutional layers (followed by an average pooling layer) as the representation network.

With Self-Supervised Learning (SimCLR): For the more challenging real-world datasets (CIFAR10, CIFAR100, ImageNet), learning generative features of images with VAE is difficult, so we use self-supervised learning methods (SimCLR) to validate our robustness regularizer. In this experiment, we consider RoCL [7] as our main baseline. We also include supervised learning models (Standard Training, AT [5] and TRADES [6]) for reference.

We train our self-supervised model with 2000 epochs (except for ImageNet, where we train a total number of 200 epochs for computational reasons). Similar to [7], we consider the l_∞ attack and defense. Following standard in the adversarial robustness literature, we set the training perturbation radius as $\epsilon = 8/255$ for CIFAR10/CIFAR100 and $\epsilon = 2/255$ for ImageNet.

For CIFAR10, we use ResNet18 [22] as the backbone network with a batchsize of 1024. As for the CIFAR100 dataset, we use ResNet50 as the backbone, and also with a batchsize of 1024. For ImageNet, we use a batchsize of 4096 with a ResNet50 network. For all experiments, we use a starting learning rate of 1.2 and perform Cosine annealing on the learning rate over the course of training.

4.3 Results

4.3.1 With VAE

Table 1 shows the results of our model and the baselines. MNIST is a relatively easy dataset, so most methods perform reasonably well. Noticeably, our model (VAE+Urkle) outperforms AE+TRADES by 1.4%, and approaches the performance of supervised adversarial training methods.

Table 1: **MNIST** with l_∞ adversaries. Training and testing ϵ are set to 0.1 in this experiment. Our method (VAE+Urkle) outperforms the baseline AE+TRADES, while approaching the robustness similar to supervised adversarial training methods.

Models	Clean Acc	Adversarial Acc
Standard Training	99.3±0.1	1.0±0.3
AT	99.0±0.1	**98.5±0.2**
TRADES	99.1±0.1	98.3±0.1
AE + TRADES	99.1±0.1	96.6±0.4
VAE + Urkle (ours)	99.1±0.1	**98.0±0.1**

Table 2: **CIFAR10**: Results of supervised and self-supervised methods trained with l_∞ adversaries and $\epsilon = 8/255$ (when applicable). Our method (SimCLR+Urkle) significantly outperforms the baseline RoCL, especially with unseen (and stronger) attack $\epsilon = 16/255$.

	CIFAR10								
	Fully Labeled Data			5000 Labeled Data			1000 Labeled Data		
Model	Clean	8/255	16/255	Clean	8/255	16/255	Clean	8/255	16/255
Standard Training	92.82	0.00	0.00	79.09	0.00	0.00	60.39	0.00	0.00
AT	81.63	44.50	14.47	64.97	24.52	6.69	50.03	15.26	3.91
TRADES	77.03	**48.01**	**22.55**	63.14	**25.97**	**7.78**	48.32	**15.92**	**3.97**
SimCLR	91.25	0.63	0.15	84.31	0.84	0.12	82.15	0.55	0.11
RoCL	83.71	40.27	9.55	78.82	36.93	9.90	76.49	34.44	8.96
SimCLR+Urkle (ours)	82.31	**42.56**	**14.29**	77.47	**38.76**	**12.94**	74.82	**37.56**	**12.22**

Table 3: **CIFAR100**: esults of supervised and self-supervised methods trained with l_∞ adversaries and $\epsilon = 8/255$ (when applicable). Our method (SimCLR+Urkle) significantly outperforms the baseline RoCL.

	CIFAR100								
	Fully Labeled Data			5000 Labeled Data			1000 Labeled Data		
Model	Clean	8/255	16/255	Clean	8/255	16/255	Clean	8/255	16/255
Standard Training	70.34	0.00	0.00	26.59	0.00	0.00	12.14	0.00	0.00
AT	52.87	**19.46**	**6.80**	**21.05**	5.30	1.52	12.46	**3.26**	0.95
TRADES	56.96	18.54	4.48	20.35	**6.41**	**1.63**	13.78	3.19	**1.32**
SimCLR	58.79	0.47	0.00	53.23	0.46	0.12	44.16	0.57	0.26
RoCL	52.19	22.00	8.35	40.60	18.86	7.25	30.23	13.83	5.55
SimCLR+Urkle (ours)	53.81	**24.82**	**10.63**	42.66	**22.34**	**11.06**	32.12	**14.40**	**6.71**

Table 4: **ImageNet** with l_∞ adversaries and $\epsilon = 2/255$.

Models	Clean Acc	Adversarial Acc
RoCL	52.46	23.19
SimCLR + Urkle (ours)	51.19	**25.69**

4.3.2 With Self-Supervised Learning (SimCLR):

CIFAR10 and CIFAR100: As aforementioned, in the CIFAR10 and CIFAR100 experiments, we train all models (except for Standard Training and SimCLR) with $\epsilon = 8/255$. We evaluate these models against l_∞ adversarial attack with strength $\epsilon = 8/255$ and $\epsilon = 16/255$. Table 2 and Table 3 show that our method (SimCLR+Urkle) clearly outperforms RoCL (with a slight trade-off of clean accuracy in some experiments), indicating the effectiveness of our robustness regularizer. Especially, our method is more robust against unseen attack strength (ϵ=16/255).

As discussed earlier, there is a trade-off between the clean accuracy and adversarial robustness in our model (as well as other adversarial training methods such as TRADES), resulting in a slightly lower (around 1%) clean accuracy of our model when compared to RoCL. However, our model outperforms RoCL significantly in terms of adversarial robustness (especially for $\epsilon = 16/255$, which is an unseen attack strength). We find that this is reasonable and the improved robustness is well worth the trade-off. Note that a similar trend can be observed for TRADES and AT, where TRADES achieves lower clean accuracy but much better adversarial robustness when compared to AT.

ImageNet: Preliminary result on the ImageNet dataset (Table 4) also indicates that our method outperforms RoCL on this large scale dataset.

Experimental Results with limited numbers of labels: To take advantage of the unsupervised nature of our method, we also conduct the experiments when the number of labeled images is limited. To be re-emphasize, we train our SSL model and the unsupervised robustness regularizer **without any labels**, and the limited number of labels are only used for the training of the task-specific output head $\hat{p}(y|z)$ (without adversarial training). Table 2 and Table 3 report the results for CIFAR10 and CIFAR100 with 5000 and 1000 labels. It can be clearly seen that supervised methods fail to learn a robust model with such a few available labels. Among the unsupervised robust representation learning methods, our model also significantly outperforms the baseline RoCL in these scenarios.

5 Conclusion

To conclude, in this paper, we develop a task-agnostic robust representation learning method. The core idea behind our method is to minimize a task-independent robustness regularizer that enforces the representation of an image to be close to that of its adversarial examples. This is motivated by our theoretical result that, for a model using the learned representation for a downstream task, its adversarial loss is bounded by the loss on clean image plus the above task-independent regularizer. Our regularizer can be straightforwardly applied to almost any existing representation learning method (with only an adaptation to a probabilistic representation). To the best of our knowledge, our work is one of the first to study the problem of unsupervised robust representation learning in a principled way, and show that the robustness can be theoretically transferred to the downstream tasks. We demonstrate our proposed regularizer with several unsupervised/self-supervised methods (from VAE to SimCLR), and conduct extensive experiments on MNIST, CIFAR10, CIFAR100 and ImageNet to validate our method. Experimental results suggest that our proposed method (when used with SimCLR) achieves SOTA performance on the unsupervised robust representation learning task.

References

[1] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[2] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021.

[4] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. *arXiv preprint arXiv:2103.03230*, 2021.

[5] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[6] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[7] Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. *arXiv preprint arXiv:2006.07589*, 2020.

[8] 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.

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

[10] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

[11] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *arXiv preprint arXiv:2007.02617*, 2020.

[12] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[13] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *NeurIPS*, 2020.

[14] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*, 2019.

[15] Jean-Baptiste Alayrac, Jonathan Uesato, Po-Sen Huang, Alhussein Fawzi, Robert Stanforth, and Pushmeet Kohli. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems*, 32, 2019.

[16] Pranjal Awasthi, Vaggos Chatziafratis, Xue Chen, and Aravindan Vijayaraghavan. Adversarially robust low dimensional representations. In *Conference on Learning Theory*, pages 237–325. PMLR, 2021.

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

[18] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. 2016.

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

[20] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[21] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[23] A Tuan Nguyen, Toan Tran, Yarin Gal, Philip HS Torr, and Atılım Güneş Baydin. Kl guided domain adaptation. *arXiv preprint arXiv:2106.07780*, 2021.

A Proofs

A.1 Proposition 1

We use the same idea as the proof of Proposition 1 in [23], which is a bound on the target loss in the domain adaptation problem. This is because our setting can somewhat be cast as a domain adaptation problem, where the source domain is the clean data distribution, and the target domain is the adversarial data distribution. The proof is as below:

Proof.
Let $a(x) = \arg\max_{x^{adv} \in A(x)} -\log \mathbb{E}_{p_\theta(z|x^{adv})}[\hat{p}_T(y|z)]\ \forall x \in \mathcal{X}$. We need to prove that:

$$\begin{aligned}\mathbb{E}_{p_T(x,y)}\left[-\log \mathbb{E}_{p_\theta(z|a(x))}[\hat{p}_T(y|z)]\right] \leq \mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right] \\ + \frac{M}{\sqrt{2}}\sqrt{\mathbb{E}_{p(x)}\left[\max_{x^{u\text{-}adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\text{-}adv})]\right]}\end{aligned} \tag{13}$$

Due to Jensen Inequality, we have:

$$\mathbb{E}_{p_T(x,y)}\left[-\log \mathbb{E}_{p_\theta(z|a(x))}[\hat{p}_T(y|z)]\right] \leq \mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|a(x))}[-\log \hat{p}_T(y|z)]\right] \tag{14}$$

Therefore, we only need to prove that:

$$\begin{aligned}&\mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|a(x))}[-\log \hat{p}_T(y|z)]\right] - \mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right] \\ \leq &\frac{M}{\sqrt{2}}\sqrt{\mathbb{E}_{p(x)}\left[\max_{x^{u\text{-}adv} \in A(x)} \mathrm{KL}[p_\theta(z|x)|p_\theta(z|x^{u\text{-}adv})]\right]}\end{aligned} \tag{15}$$

We have:

$$\begin{aligned}
&\mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|a(x))}[-\log \hat{p}_T(y|z)]\right] - \mathbb{E}_{p_T(x,y)}\left[\mathbb{E}_{p_\theta(z|x)}[-\log \hat{p}_T(y|z)]\right] && (16)\\
=&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{Z}} p_\theta(z|a(x))[-\log \hat{p}_T(y|z)]dz - \int_{\mathcal{Z}} p_\theta(z|x)[-\log \hat{p}_T(y|z)]dz\right] && (17)\\
=&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{Z}} -\log \hat{p}_T(y|z)\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right] && (18)
\end{aligned}$$

For all x, let $\mathcal{A}(x) = \{z \in \mathcal{Z} | p_\theta(z|a(x)) - p_\theta(z|x) \geq 0\}$ and $\mathcal{B}(x) = \{z \in \mathcal{Z} | p_\theta(z|a(x)) - p_\theta(z|x) < 0\}$, then:

$$\begin{aligned}
&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{Z}} -\log \hat{p}_T(y|z)\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right] && (19)\\
=&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{A}(x)} -\log \hat{p}_T(y|z)\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right.\\
&\left.+\int_{\mathcal{B}(x)} -\log \hat{p}_T(y|z)\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right] && (20)\\
\leq&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{A}(x)} -\log \hat{p}_T(y|z)\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right] && (21)\\
&\text{(since } -\log \hat{p}_T(y|z) \text{ is a non-negative quantity)}\\
\leq&\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{A}(x)} M\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz\right] && (22)\\
=&M\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{A}(x)} \left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz\right] && (23)
\end{aligned}$$

We have:

$$\begin{aligned}
&\int_{\mathcal{Z}}\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz = 0 && (24)\\
\Rightarrow&\int_{\mathcal{A}(x)}\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz + \int_{\mathcal{B}(x)}\left[p_\theta(z|a(x)) - p_\theta(z|x)\right]dz = 0 && (25)\\
\Rightarrow&\int_{\mathcal{A}(x)}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz = \int_{\mathcal{B}(x)}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz && (26)\\
\Rightarrow&\int_{\mathcal{A}(x)}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz = \frac{1}{2}\int_{\mathcal{Z}}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz && (27)
\end{aligned}$$

Due to the Pinsker's Inequality we have:

$$\begin{aligned}
&\frac{1}{2}\int_{\mathcal{Z}}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz\\
\leq&\frac{1}{2}\sqrt{2\int_{\mathcal{Z}} p_\theta(z|x)\log\frac{p_\theta(z|x)}{p_\theta(z|a(x))}dz} && (28)\\
=&\frac{1}{\sqrt{2}}\sqrt{\text{KL}[p_\theta(z|x)|p_\theta(z|a(x))]} && (29)
\end{aligned}$$

Therefore:

$$\begin{aligned}
&M\mathbb{E}_{p_T(x,y)}\left[\int_{\mathcal{A}(x)}\left|p_\theta(z|a(x)) - p_\theta(z|x)\right|dz\right] && (30)\\
\leq&\frac{M}{\sqrt{2}}\mathbb{E}_{p_T(x,y)}\left[\sqrt{\text{KL}[p_\theta(z|x)|p_\theta(z|a(x))]}\right] && (31)\\
=&\frac{M}{\sqrt{2}}\mathbb{E}_{p(x)}\left[\sqrt{\text{KL}[p_\theta(z|x)|p_\theta(z|a(x))]}\right] && (32)\\
\leq&\frac{M}{\sqrt{2}}\sqrt{\mathbb{E}_{p(x)}\left[\text{KL}[p_\theta(z|x)|p_\theta(z|a(x))]\right]} && (33)\\
\leq&\frac{M}{\sqrt{2}}\sqrt{\mathbb{E}_{p(x)}\left[\max_{x^{u\_adv}\in A(x)}\text{KL}[p_\theta(z|x)|p_\theta(z|x^{u\_adv})]\right]} && (34)
\end{aligned}$$

We conclude our proof. □

B Experimental Results

B.1 Details on the baseline AE+TRADES

Here we describe the baseline AE+TRADES, which we use in the VAE experiment, in more detail. In particular, an autoencoder (AE) consists of an encoder g_θ and a decoder h_ϕ. The encoder g transforms the input x to a representation z (often lower dimensional), i.e., $z = g_\theta(x)$; while the decoder h tries to reconstruct the original input from the representation, i.e., $\hat{x} = h_\phi(z)$. Using the mean squared (l2) distance for the reconstruction, the objective of AE is:

$$\mathbb{E}_{p(x)}[||x - h_\phi(g_\theta(x))||_2^2] \tag{35}$$

Since the reconstruction $h \circ g$ can be treated as a prediction task (predicting the original x), we can use a TRADES-like regularizer to make the model more robust, and thus the encoder is also more robust. Specially, we can enforce the reconstruction of an image to be similar to that of its adversaries. The final objective is:

$$\begin{aligned}&\mathbb{E}_{p(x)}[||x - h_\phi(g_\theta(x))||_2^2] \\ &+ \beta\mathbb{E}_{p(x)}[\max_{x^{adv}\in A(x)} ||h_\phi(g_\theta(x^{adv})) - h_\phi(g_\theta(x))||_2^2]\end{aligned} \tag{36}$$

Note that we can only use this baseline with AE (not VAE) because the TRADES regularizer only works straightforwardly with a deterministic model / prediction.

B.2 Experimental Settings

B.2.1 With VAE

For the VAE MNIST experiment, the encoder (representation network) is a simple convolutional network with 4 `3x3` convolutional layers (with the last layer has 128 channels so that the representation has 128 dimension), followed by an average pooling layer.

With AE and VAE, the decoder consists of 4 ConvTranspose2d layers, mirroring the encoder.

The classifier (from a representation to the prediction label) is a composition of 3 fully connected layers (with batchnorm and ReLU activation in-between).

B.2.2 With SimCLR

For a ResNet18 backbone network, we set the representation dimension to 512, while that for a ResNet50 backbone is 2048. We set the initial learning rate to 1.2 and do Cosine annealing to 0. Other experiment details have been presented in the main paper. In addition, please also refer to our code for more details.