RUSH: Robust Contrastive Learning via Randomized Smoothing

Yijiang Pang [1] **Boyang Liu** [1] **Jiayu Zhou** [1]

Abstract

Recently, adversarial training has been incorporated in self-supervised contrastive pre-training to augment label efficiency with exciting adversarial robustness. However, the robustness came at a cost of expensive adversarial training. In this paper, we show a surprising fact that contrastive pre-training has an interesting yet implicit connection with robustness, and such natural robustness in the pre-trained representation enables us to design a powerful robust algorithm against adversarial attacks, RUSH, that combines the standard contrastive pre-training and randomized smoothing. It boosts both standard accuracy and robust accuracy, and significantly reduces training costs as compared with adversarial training. We use extensive empirical studies to show that the proposed RUSH outperforms robust classifiers from adversarial training, by a significant margin on common benchmarks (CIFAR-10, CIFAR-100, and STL-10) under first-order attacks. In particular, under ℓ_∞-norm perturbations of size 8/255 PGD attack on CIFAR-10, our model using ResNet-18 as backbone reached 77.8% robust accuracy and 87.9% standard accuracy. Our work has an improvement of over 15% in robust accuracy and a slight improvement in standard accuracy, compared to the state-of-the-arts. Full version is available.

1. Introduction

Adversarial attacks have imposed a significant challenge to inference robustness of machine learning models. For example, an image classifier that achieves high accuracy for normal testing data could be easily fooled by a perturbed image from adversarial attacks, and the perturbed images usually have very subtle differences from normal images (Goodfellow et al., 2014; Akhtar & Mian, 2018). To tackle this challenge, many defense mechanisms are proposed to achieve adversarial robustness (Kannan et al., 2018; Madry et al., 2017; Salman et al., 2019; Jiang et al., 2020). The defense algorithms can be categorized into empirical defense and certified defense, where the latter provides provable robust guarantees (i.e. there exist no adversarial samples given bounded perturbation). One popular empirical defense is adversarial training (AT), which augments training data with adversarial examples. AT is usually achieved by alternative min-max optimization (Goodfellow et al., 2014; Kurakin et al., 2016). However, failed adversarial examples from unstable attacks may break the defense, as a result of numerical instability, randomness, and vanishing/exploding gradients (Athalye et al., 2018).

On the other hand, certified defenses provide provable robustness that the predictions of classifier remain constant within certain neighborhoods. One class of approaches reach certified robustness through robust training, which are guaranteed to detect all adversarial examples around a data point in a norm-bounded range by constructing an upper bound on the worst-case loss through convex relaxations (Raghunathan et al., 2018; Wong & Kolter, 2018). Due to large computational costs, those methods are mostly restricted to applications of small-scale data. Randomized smoothing provides certified defenses for large datasets. It smooths a classifier by applying the convolution between Gaussian noise and the classifier, and is certifiably robust to adversarial perturbations under the ℓ_2 norm (Cohen et al., 2019; Salman et al., 2019). Although certificated defense can provide us theoretical guarantees against the adversarial attack, compared with the empirical defense such as adversarial training, the empirical performance of certificated defense is usually worse than empirical defenses.

Introducing robustness in Self-Supervised Learning (SSL) recently has attracted significant efforts from the community (Hendrycks et al., 2019; Alayrac et al., 2019; Gowal et al., 2020), as recent years witnessed its huge success in many applications that dramatically improves generalization performance by leveraging unlabeled data for downstream tasks. A typical SSL treatment includes a self-supervised contrastive pre-training stage, followed by a supervised fine-tuning stage that tailors the pre-trained model to supervisions (Jing & Tian, 2020; He et al., 2020; Chen et al., 2020a;

[1]Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48823, USA. Correspondence to: Jiayu Zhou <jiayuz@msu.edu>.

Workshop on New Frontiers in Adversarial Machine Learning at ICML 2022, Copyright 2022 by the author(s).

Grill et al., 2020). Unfortunately, recent studies showed that SSL does not spare from robustness issues (Hendrycks et al., 2019; Alayrac et al., 2019; Gowal et al., 2020). Attempts of integrating adversarial training in SSL have demonstrated improving robustness by paying the price of expensive computation costs that can be forbidden in many resource-limited data mining applications (Chen et al., 2020b; Jiang et al., 2020).

As such, in this paper, we ask the following intriguing and important question: *is adversarial training necessary to achieve robustness in contrastive pre-training?* By studying the similarity between Lipschitz continuity and optimization objective of contrastive learning (c.f. Section 4.1), we reveal the *natural robustness* introduced by contrastive learning. With such robustness, we propose an embarrassingly simple framework to deliver powerful robustness by combining contrastive learning and randomized smoothing. We demonstrate that the model trained with the proposed approach, which follows the SimCLR training procedures, shows significant robustness to Projected Gradient Decent (PGD) attack (Madry et al., 2017) and AutoAttack (AA) (Croce & Hein, 2020) on CIFAR-10, CIFAR-100, and STL10, achieving state-of-art accuracy for clean data and perturbed data (Krizhevsky et al., 2009; Coates et al., 2011).

Our contributions. Our paper has the following contributions. Firstly, we discuss the potential connection between Lipschitz continuity and contrastive pre-training, which leads to the discovery of hidden natural robustness induced by contrastive learning. Secondly, our proposed approach, which strategically combines the natural robustness with randomized smoothing, has shown significant robustness against first-order-based attacks with a very limited sacrifice of the standard accuracy. Finally, we conducted extensive empirical studies to evaluate the proposed algorithm, which delivers comparable results against state-of-the-arts while enjoys much smaller training costs.

2. Related Work

Adversarial Training. Existing literature studied the adversarial robustness of neural networks (Moosavi-Dezfooli et al., 2016; Papernot et al., 2016; Carlini & Wagner, 2017; Xu et al., 2017). Adversarial training (AT) is one of the most powerful methods against adversarial attacks (Madry et al., 2017), which injects adversarial examples in the original training set. AT is usually associated with huge computational overhead, leading to efforts on data-efficient and computation-efficient variants to mitigate the cost of the generating of adversarial examples (Shafahi et al., 2019; Wong et al., 2020; Zhang et al., 2019b). However, these approaches still demand considerable additional computational costs for constructing and augmenting adversarial examples as compared to the standard training.

Adversarial Training in Self-supervised Learning. Recently, adversarial examples were shown to improve the robustness of self-supervised learning (Hendrycks et al., 2019). Continued efforts developed a variety of approaches to introduce adversarial training in the context of self-supervised learning (Alayrac et al., 2019; Chen et al., 2020b; Gowal et al., 2020). A prominent example is SimCLR, a popular and simple contrastive learning framework with exceptional performance (Chen et al., 2020a). Injecting adversarial samples to the pre-training stage of SimCLR is shown to successfully induce model robustness (Jiang et al., 2020). Alternatively, robustness can be achieved by a training process that maximizes the similarity between a random augmentation of a data sample and its instance-wise adversarial perturbation (Kim et al., 2020). Most existing robust approaches in contrastive learning rely on adversarial training, which is associated with significant computational overhead and decreased standard accuracy, and often provides no provable guarantees.

Certified Defenses. In contrast to empirical approaches for robustness such as adversarial training, certified defense guarantees provable robustness to a specific class of adversarial perturbation (Katz et al., 2017; Tjeng et al., 2017; Gehr et al., 2018; Mirman et al., 2018; Raghunathan et al., 2018; Weng et al., 2018; Wong & Kolter, 2018). Many existing approaches admit efficiency issues and are hard to scale to large datasets (Salman et al., 2019; Cohen et al., 2019). Randomized smoothing, a kind of certified defenses, was developed following the idea of augmenting the dataset by sampling some data points centered at the original data examples in the input space to against adversarial perturbation (Liu et al., 2018; Cao & Gong, 2017), but no guarantees have been proved in these works. Following works largely expanded the generalization of randomized smoothing and proved its probabilistic robustness guarantee against ℓ_p-bounded adversaries (Cohen et al., 2019; Teng et al., 2019; Zhang et al., 2019a; Levine & Feizi, 2020). A recent work comprehensively studied the interaction between two important factors of random smoothing: the choice of smoothing distribution and the perturbation norm(Yang et al., 2020). Combining adversarial training with randomized smoothing classifiers is recently shown to substantially boost certified robustness (Salman et al., 2019), and it suffers the aforementioned issues due to its usage of adversarial training. Besides, the accuracy trade-off has also been observed in randomized smoothing that adversarially trained networks with higher robust accuracy tend to have lower standard accuracy (Tsipras et al., 2018; Cohen et al., 2019).

3. Notations and Preliminaries

In this section, we introduce notations and revisit preliminaries, including contrastive learning, randomized smoothing,

and adversarially perturbed data. We use $(\mathbf{x}_i, \mathbf{y}_i)$ to denote one single labeled data point or an observation, and the training set $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^k$.

Contrastive Learning is a technique that uses unlabeled data to learn feature representations by constructing supervisions of similarity/dissimilar pairs from data. SimCLR is a widely-adopted contrastive learning framework (Chen et al., 2020a) due to its high performance and simplicity. We will use SimCLR as our backbone of contrastive learning for discussion in this paper.

Formally, SimCLR has two stages: self-supervised pre-training and supervised fine-tuning. During the pre-training stage, it leverages a set of pre-defined data transformations $\mathcal{T}$, samples two transformations $t_1, t_2 \sim \mathcal{T}$, and uses them to transform a given data point $\mathbf{x}$ into two views $t_1(\mathbf{x}), t_2(\mathbf{x})$. It then uses a multi-view contrastive learning loss $\mathcal{L}_{pre}$ to learn a feature extractor and projector $g_{\theta_p} \circ f_\theta$, where the model equipping with projector $g(\cdot)$, parameterized by θ_p, on top of the feature extractor $f(\cdot)$, parameterized by θ. The loss maximizes the cosine similarity of the projected data representations of different views of the same image and minimizes the cosine similarity of the projected data representations generated by different images. In the fine-tuning stage, it uses the labeled training data in $\mathcal{D}$ to learn a classifier $\phi(\cdot)$, parameterized by θ_c, with representations from the feature extractor $f(\cdot)$. The objectives are given by:

PRE-TRAIN:

$$\min_{\theta,\theta_p} \mathbb{E}_{t_1,t_2\sim\mathcal{T},\mathbf{x}\in\mathcal{D}} \mathcal{L}_{pre}(g_{\theta_p} \circ f_\theta([t_1(\mathbf{x}), t_2(\mathbf{x})]));$$

FINE-TUNE:

$$\min_{\theta_c} \mathbb{E}_{\mathbf{x},\mathbf{y}\in\mathcal{D}} \mathcal{L}_{CE}(\phi_{\theta_c} \circ f_\theta(\mathbf{x}), \mathbf{y}).$$

Randomized Smoothing aims to construct a smooth classifier G from a base classifier $F : \mathcal{X} \in \mathbb{R}^d \to \mathcal{Y} \in \mathbb{R}$ that maps an instance space $\mathcal{X}$ to an output space $\mathcal{Y}$. The smoothed classifier G responds to a query $\mathbf{x}$ with the class that the base classifier F is most likely to return when $\mathbf{x}$ is perturbed by isotropic Gaussian noise (Cohen et al., 2019; Salman et al., 2019):

$$G(\mathbf{x}) = (F * \mathcal{N}_d(0, \sigma^2))(x) = \mathop{\mathbb{E}}_{\delta\sim\mathcal{N}_d(0,\sigma^2)}[F(\mathbf{x} + \delta)], \tag{1}$$

where σ denotes noise level to trade-off between robustness and accuracy.

There are several approaches to train a smoothed classifier (Cohen et al., 2019; Salman et al., 2019; Lecuyer et al., 2019). In this paper, we use the following representative method, one-time-noise-based augmentation, to train our classifier:

$$\mathcal{L}(\mathbf{x}, \mathbf{y}) = -\log F(\mathbf{x} + \delta)_{\mathbf{y}}, \tag{2}$$

where δ is sampled from smoothing distribution. The choice of smoothing distribution including its noise scale is key to empirical success of random smoothing (Yang et al., 2020). We denote the randomized smoothing operator to data $\mathbf{x}$ as $\mathcal{Q} : \mathbb{R}^d \to \mathbb{R}^d$:

$$\mathcal{Q}(\mathbf{x}) = \mathbf{x} + \delta, \tag{3}$$

where $\delta \in \mathbb{R}^d \sim \Omega_d$ is a vector sampled from a certain smoothing distribution Ω_d. Common choices of smoothing distributions include uniform distribution $\delta \sim U_d(-\mu, \mu)$ and Gaussian distribution $\delta \sim \mathcal{N}_d(0, \sigma^2)$. A comprehensive study of various smoothing distributions is provided in (Yang et al., 2020).

During the inference, randomized smoothing uses Monte-Carlo sampling to estimate the expectation in equation 1. Similarly, we use a voting strategy, denoted by $\mathcal{V}_m(F(\mathbf{x}))$, to estimate the expected prediction $\hat{y}$ for an input $\mathbf{x}$, from outputs generated by m times randomly smoothed $\mathbf{x}$:

$$\hat{y} = \mathop{\text{VOTING}}_{i=\{1,\ldots,m\}} (F(\mathcal{Q}_i(\mathbf{x})) \tag{4}$$

$$= \mathop{\text{VOTING}}_{i=\{1,\ldots,m\}} (F(\mathbf{x} + \delta_i)) \equiv \mathcal{V}_m(F(\mathbf{x})), \tag{5}$$

where m is a hyper-parameter. A relatively larger m will give a more accurate estimation for the expectation. Besides voting for prediction, one can also average the confidence scores from the underlying classifier for each class (Kumar et al., 2020).

Adversarially Perturbed Data is used to conduct adversarial attacks and test model robustness (Goodfellow et al., 2014). Given an observation $(\mathbf{x}, \mathbf{y})$ and a trained model G, the goal is to find a point $\hat{\mathbf{x}}$ that has a small difference with $\mathbf{x}$ and misleads G to an incorrect prediction:

$$\hat{\mathbf{x}} = \text{argmax}_{||\mathbf{x}'-\mathbf{x}||_p\leq\epsilon} \ell_{CE}(G(\mathbf{x}'), \mathbf{y}), \tag{6}$$

where ϵ is the scale of ℓ_p-norm perturbation. Directly attacking one data point $(\mathbf{x}, \mathbf{y})$ without augmented smoothing operation is shown to have ill-behaved problem under randomized smoothing modeling conditions due to the argmax operator (Salman et al., 2019; Cohen et al., 2019). We use a similar strategy to generate the adversarial samples against smoothed classifier (Salman et al., 2019).

Specifically, let $J(\mathbf{x}') = \ell_{CE}(G(\mathbf{x}'), \mathbf{y}))$ be the objective function in equation 6, and G is the smoothed classifier, the empirical gradient of finding $\hat{x}$ is given by the formula based on Monte Carlo approximations (Salman et al., 2019):

$$\nabla_{\mathbf{x}'} J(\mathbf{x}') \approx \nabla_{\mathbf{x}'} \left(-\log \left(\frac{1}{m} \sum\nolimits_{i=1}^{m} F(\mathbf{x}' + \delta_i)_{\mathbf{y}} \right) \right), \tag{7}$$

where m is the same hyper-parameter denoting the number of voting samples in equation 4 as well as the number of

Algorithm 1 RUSH: robust contrastive learning via randomized smoothing

Require:
$\mathcal{D}, \mathcal{D}_{\text{test}}$: training and test datasets
$f_\theta, g_{\theta_p}, \phi_{\theta_c}$: randomly initialized feature extractor, projector, and linear classifier
$\mathcal{Q}$: data smoothing operation
$\mathcal{A}_m^+$: adversarially perturbing operation
$\mathcal{V}_m$: voting strategy of randomized smoothing
$\mathcal{T}$: pre-defined data transformation
Training: pre-training $\rightarrow$ fine-tuning
repeat
  for $(\mathbf{x}, \mathbf{y})$ **in** $\mathcal{D}$ **do**
    $\mathbf{x}_1, \mathbf{x}_2 = t_1(\mathbf{x}), t_2(\mathbf{x}),$ where $t_1, t_2 \sim \mathcal{T}$
    $\mathbf{x}_2 = \mathcal{Q}(\mathbf{x}_2)$
    $(\theta, \theta_p) \leftarrow (\theta, \theta_p) - \alpha \nabla_{\theta, \theta_p} \mathcal{L}_{pre}(g_{\theta_p} \circ f_\theta([\mathbf{x}_1, \mathbf{x}_2]))$
  end for
until pre-training stop
repeat
  for $(\mathbf{x}, \mathbf{y})$ **in** $\mathcal{D}$ **do**
    $\mathbf{x} = \mathcal{Q}(\mathbf{x})$
    $\theta_c \leftarrow \theta_c - \beta \nabla_{\theta_c} \mathcal{L}_{CE}(\phi_{\theta_c} \circ f_\theta(\mathbf{x}), \mathbf{y})$[1]
  end for
until fine-tuning stop
Inference: Given Input $\mathbf{x}$
$\hat{\mathbf{y}} = \mathcal{V}_m(\phi_{\theta_c} \circ f_\theta(\mathbf{x}))$
return $\hat{\mathbf{y}}$

Notes:
[1] The bias of the BatchNorm layers of f_θ is not frozen during fine-tuning, since batch normalization should be based on specific feature distributions, especially in the transfer learning setting.

samples of Monte Carlo approximations in equation 7. We denote adversarial perturbation from equation 6 by employing Monte Carlo approximation, equation 7, as $\mathcal{A}_m^+(\mathbf{x}) = \hat{\mathbf{x}}$.

Without ambiguity, our approach only uses adversarially perturbed data $\mathcal{A}_m^+(\mathbf{x})$ in the inference stage to evaluate robust performance. In this study, PGD attack (Madry et al., 2017) and its variants (Croce & Hein, 2020) are used to get the practically perturbed data.

4. Robust Contrastive Learning via Randomized Smoothing

In this section, we first discuss the foundation of RUSH, an interesting yet implicit connection between contrastive learning and robustness, which suggests the existence of the *natural robustness* of contrastive learning under standard training. Such natural robustness allows us to propose a simple, yet extremely efficient and effective robust contrastive learning via randomized smoothing (RUSH). RUSH has a two-staged training process: self-supervised pre-training and supervised fine-tuning. During the pre-training stage, it uses standard training of contrastive learning, with basic noise-based data augmentation. The following fine-tuning stage and inference are empowered by the randomized smoothing paradigm, where we arm the linear classifier with label information and robust feature representation to defense adversarial attacks. The proposed RUSH algorithm is illustrated in Algorithm 1.

4.1. Natural robustness in contrastive learning

We now illustrate a key property of contrastive learning, i.e., the *natural robustness*, as a result of the implicit connection between Lipschitz continuity and the objective of contrastive learning. To see this, we will first elaborate how robustness is related to Lipschitz continuous, and then show why contrastive learning is impacting Lipschitz continuous. In the following, let f be the feature encoder of a classifier following contrastive learning framework which maps inputs $\mathcal{X} \in \mathbb{R}^d$ to feature representation in $\mathcal{Z} \in \mathbb{R}^p$. Recall that $\mathcal{T}$ denotes a set of pre-defined data transformations in contrastive learning, $d_{\mathbf{x}}$ and $d_{\mathbf{z}}$ denote distance metrics measuring $\mathbf{x}, \mathbf{z}$.

Lipschitz continuity and Robustness. According to definition, if f is Lipschitz continuous, then we have the following:

$$d_{\mathbf{z}}(f(\mathbf{x}_i), f(\mathbf{x}_j)) \leq K \cdot d_{\mathbf{x}}(\mathbf{x}_i, \mathbf{x}_j), \tag{8}$$

where K is referred to as a Lipschitz constant. Since the definition holds for any $\mathbf{x}_i$, $\mathbf{x}_j$, we could let $\mathbf{x}_j$ be the adversarial sample of $\mathbf{x}_i$. Given an original sample $\mathbf{x}_i$, its adversarial sample $\text{adv}(\mathbf{x}_i)$ is designed to be as close to the original as possible. Therefore, we can assume the distance between the adversarial sample and the original sample is bounded by c, and we thus have:

$$d_{\mathbf{z}}(f(\mathbf{x}_i), f(\text{adv}(\mathbf{x}_i))) \leq Kc,$$

from which we see that if a learned function or model has a small Lipschitz constant K, then the model is robust against adversarial attacks since the change in output is also bounded given small K.

Contrastive learning and Lipschitz continuity. Now recall the goal of contrastive learning, which encourages the learned feature representation from f should be similar if the input pair of images have similar content. As such, given a data point $\mathbf{x}$, contrastive learning in the pre-training stage embeds the following optimization objective:

$$\min_f \mathbb{E}_{\mathbf{x}_1, \mathbf{x}_2 \sim \Gamma} d_{\mathbf{z}}(f(\mathbf{x}_1), f(\mathbf{x}_2)), \tag{9}$$

where $\Gamma = \{d_{\mathbf{x}}(\mathbf{x}_1, \mathbf{x}_2) \leq c\}$ is the set of arbitrary pairs of two data points, as long as the distance between them is smaller than c. To optimize the above objective, the key question is how we get the samples from the set Γ. For example, the contrastive learning backbone of this paper (SimCLR) used a pair of transformations $t_1(\mathbf{x}), t_2(\mathbf{x})$ as the samples from Γ, where $t_1, t_2 \sim \mathcal{T}$. Therefore, we have the following objective:

$$\min_f \mathbb{E}_{\mathbf{x}_1 \sim t_1(\mathbf{x}), \mathbf{x}_2 \sim t_2(\mathbf{x})} d_{\mathbf{z}}(f(\mathbf{x}_1), f(\mathbf{x}_2)), \forall t_1, t_2 \sim \mathcal{T}. \tag{10}$$

We would like to highlight that the $d_{\mathbf{x}}$ should be treated as the distance in semantic space for the above objective. This is because although some common distance metrics for $d_{\mathbf{x}}$ such as ℓ_2-distance cannot satisfy the distance constraint for $d_{\mathbf{x}}(t_1(\mathbf{x}), t_2(\mathbf{x}))$ for certain transformations such as rotation, there *are* many other valid metrics such as Wasserstein distance that satisfy the constraint.

By minimizing the objective in equation 9, we require the difference between representations to be small given similar inputs, and therefore, it is not hard to see that we are heuristically minimizing the Lipschitz constant K in equation 8. As a result, by getting a small Lipschitz constant in contrastive learning, the learned model gains robustness against adversarial attacks. And we call such robustness as *natural robustness* since it does not require additional adversarial training.

4.2. Combining Randomized Smoothing and Contrastive Learning

The natural robustness in contrastive learning delivers adversarial robustness for representation and yet we need to ensure that final classifier based on the representation is also robust. We propose integrating randomized smoothing to achieve this goal in RUSH, taking advantage of its efficiency and certified robustness. In this section, we elaborate on how the combination is done in RUSH at the pre-training, fine-tuning, and inference stages.

Pre-training. The first step is to integrate the data smoothing operation in the pre-training stage. In this end, RUSH uses the following objective function for pre-training:

$$\min_{\theta, \theta_p} \mathbb{E}_{t_1, t_2 \sim \mathcal{T}, \mathbf{x} \in \mathcal{D}} \mathcal{L}_{pre}(g_{\theta_p} \circ f_\theta([t_1(\mathbf{x}), \mathcal{Q}(t_2(\mathbf{x}))]), \tag{11}$$

where $\mathcal{L}_{pre}$ is NT-Xent loss following the original definition from (Chen et al., 2020a), and $\mathcal{Q}$ is our selected randomized smoothing operation to the data as in equation 3, which can be considered as entity-wise noising the input. We note that different from the original SimCLR, there is no need to apply Gaussian blur in $\mathcal{T}$, since we already have our data smoothing operation $\mathcal{Q}$. More details can be found at Appendix A.

Fine-tuning. Next, we embed data smoothing operation into fine-tuning stage to extract the robust feature representations. As such, RUSH used the following objective for the fine-tuning stage:

$$\min_{\theta_c} \mathbb{E}_{\mathbf{x}, \mathbf{y} \in \mathcal{D}} \mathcal{L}_{CE}(\phi_{\theta_c} \circ f_\theta(\mathcal{Q}(\mathbf{x})), \mathbf{y}), \tag{12}$$

where $\mathcal{Q}$ is the same data smoothing operation. Unless otherwise specified, we use the same smoothing distribution as the one used for pre-training, fine-tuning and inference.

Inference. Besides the treatments in training, RUSH uses adversarial perturbation $\mathcal{A}_m^+$ in equation 7 and voting strategy $\mathcal{V}_m$ in equation 4, as illustrated in Algorithm 1.

We note that in typical randomized smoothing settings, the value of m is set to a reasonable value (e.g., 32, 64, or 128) and sometimes it varies across samples for the fixed certificated radius. From our empirical study, we find that RUSH also delivers rather promising performance when $m = 1$. Since $m = 1$ represents the situation of maximum computational efficiency of our approach. Under this inference strategy, RUSH has no extra computational overhead as compared to the vanilla SimCLR algorithm in both training and inference stages. So, unless otherwise specified, there are two test strategies in this study, $\mathcal{I}_F^+$ and $\mathcal{I}_F$, with different hyper-parameter m. Specifically, $\mathcal{I}_F$ denotes using $m = 1$ for $\mathcal{A}_m^+$ and $\mathcal{V}_m$. $\mathcal{I}_F^+$ denotes using $m > 1$ for $\mathcal{A}_m^+$ and $\mathcal{V}_m$, e.g., m = 64.

5. Experiments

In this section, we demonstrate the effectiveness of our proposed RUSH through performance comparison between RUSH and other key related approaches including cross-datasets and cross-tasks robustness against PGD attacks (Madry et al., 2017) and AA attacks (Croce & Hein, 2020);

5.1. Setup

The RUSH algorithm can be applied to a variety of machine learning and data mining problems with general structured data. As many competing approaches used image datasets for evaluation, we will follow this evaluation scheme. Specifically, we conduct experiments on three image benchmark datasets, CIFAR-10, CIFAR-100, and STL10, with two robustness evaluation metrics, standard accuracy over original images without perturbations and robust accuracy over adversarially perturbed images via first-order methods including PGD attack and AA attack.

We use ResNet-18 for the encoder architecture of f_θ, a two-layer MLP projection head g_{θ_p} to project the representation, and one linear layer for linear classifier ϕ_{θ_c}. For image transformation $\mathcal{T}$, we use random resized crop, random horizontal flip, and color distortions. Detailed implementations

are provided in Appendix A. Unless otherwise specified, we use uniform distribution $\delta \sim U_{32\times32}(-0.3, 0.3)$ as our default smoothing distribution. See Appendix B for details of training setting.

Inference Setting. Standard accuracy and robust accuracy are evaluated with clean data and perturbed data respectively under two inference strategies described in Section 4.2. Unless otherwise specified, we use $m = 64$ for the inference strategy $\mathcal{I}_F^+$, i.e., 64 number of sampling for both adversarial perturbation $\mathcal{A}_m^+$ and voting strategy $\mathcal{V}_m$. $\mathcal{I}_F$ is always with $m = 1$.

Without ambiguity, inference strategy $\mathcal{I}_F^+$ and $\mathcal{I}_F$ with adversarial perturbation $\mathcal{A}_m^+$ are designed for smoothed classifiers with stronger attack strength to our model (Salman et al., 2019). During the inference of our approach, the attacks to our model including PGD attack and AA attack are re-implemented with noise-based augmentation according to equation 7 (Kim, 2020). See Appendix C.

5.2. Performance

Overall cross-datasets and cross-tasks performance. Table 1 summarizes the performance of our approach under inference strategy $\mathcal{I}_F^+$ and $\mathcal{I}_F$ and key competing baselines over CIFAR-10, CIFAR100, CIFAR-10→STL-10, and STL-10 unlabeled→STL-10, where → denotes transfer learning setting, e.g., CIFAR-10→STL-10 means the model is pre-trained on the training set of CIFAR-10, fine-tuned on the training set of STL-10, and tested on the test set of STL-10.

We see that the proposed RUSH outperforms all competing baselines w.r.t. both standard accuracy and robust accuracy under network architecture ResNet-18 in all settings. Compared with state-of-the-arts using network architecture WideResNet-70-16, RUSH still outperforms them with a significant margin on robust accuracy. The method SmoothAdv Training uses smoothed classifier to defend first-order-based attacks (Salman et al., 2019), which uses a similar intuition as ours. However, RUSH outperforms it with less complexity of network architecture and regardless of adversarial examples.

6. Conclusions

In this paper, we designed a robust contrastive learning algorithm RUSH by exploring the natural robustness in contrastive learning. We showed that RUSH maintains a high standard accuracy and substantially improves robust accuracy. Extensive empirical studies showed that RUSH outperforms all existing ℓ_∞-robust methods by a significant margin on common image benchmarks including CIFAR-10, CIFAR-100, and STL-10.

Table 1. Performance comparison over baselines, in terms of standard accuracy, robust accuracy under ℓ_∞=8/255 PGD and ℓ_∞=8/255 AA attacks with ResNet-18 on CIFAR-10, CIFAR-100, CIFAR-10→STL-10, and $\overline{\text{STL-10}}$ →STL-10. The top performance is highlighted in bold.

Methods	Acc.(%)		
	Standard	PGD	AA
CIFAR-10			
RUSH$^{\mathcal{I}_F^+}$ (ours)	**87.9**	**77.8**	**79.5**
RUSH$^{\mathcal{I}_F}$ (ours)	86.4	73.7	74.7
RoCL (Kim et al., 2020)	83.7	40.2	-
Free-m (Shafahi et al., 2019)	85.9	46.8	-
AdvCL (Fan et al., 2021)	83.6	52.7	49.7
SmoothAdv (Salman et al., 2019)[4]	86.2	68.2	-
ACL (Jiang et al., 2020)[1]	82.5	52.8	49.3
Data AUG (Rebuffi et al., 2021)[1]	83.5	59.9	57.0
Data AUG (Rebuffi et al., 2021)[2]	92.2	-	66.5
CIFAR-100			
RUSH$^{\mathcal{I}_F^+}$ (ours)	**57.1**	**46.6**	-
RUSH$^{\mathcal{I}_F}$ (ours)	55.3	42.5	40.2
AdvCL (Fan et al., 2021)	56.7	28.7	24.7
Data AUG (Rebuffi et al., 2021)[1]	56.9	32.0	28.5
Data AUG (Rebuffi et al., 2021)[2]	63.5	-	34.6
CIFAR-10→STL-10			
RUSH$^{\mathcal{I}_F^+}$ (ours)	**73.7**	**53.3**	**55.9**
RUSH$^{\mathcal{I}_F}$ (ours)	72.0	49.4	53.4
BYORL (Gowal et al., 2020)	-	24.1	-
AdvCL (Fan et al., 2021)	63.5	37.7	34.7
$\overline{\text{STL-10}}$[3] →STL-10			
RUSH$^{\mathcal{I}_F^+}$ (ours)	76.7	64.6	67.0
RUSH$^{\mathcal{I}_F}$ (ours)	75.1	61.5	60.2

[1] using the provided pre-trained ResNet18 model and testing on our pipeline.
[2] latest state-of-the-art according to (Croce et al., 2020) (Link: https://robustbench.github.io/index.html), which applied network architecture WideResNet-70-16. The standard accuracy marked as [2] will not be ranked considering fair comparisons.
[3] $\overline{\text{STL-10}}$ denotes the independent set of unlabeled examples in STL-10.
[4] under ℓ_∞=2/255 PGD attack with ResNet-110.

Acknowledgements

This research was supported by NIH 1RF1AG072449, ONR N00014-20-1-2382, and NSF IIS-1749940.

References

Akhtar, N. and Mian, A. Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430, 2018.

Alayrac, J.-B., Uesato, J., Huang, P.-S., Fawzi, A., Stanforth, R., and Kohli, P. Are labels required for improving adversarial robustness? *Advances in Neural Information Processing Systems*, 32, 2019.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018.

Cao, X. and Gong, N. Z. Mitigating evasion attacks to deep neural networks via region-based classification. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pp. 278–287, 2017.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pp. 39–57. IEEE, 2017.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, T., Liu, S., Chang, S., Cheng, Y., Amini, L., and Wang, Z. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 699–708, 2020b.

Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pp. 1310–1320. PMLR, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020.

Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

Fan, L., Liu, S., Chen, P.-Y., Zhang, G., and Gan, C. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? *Advances in Neural Information Processing Systems*, 34, 2021.

Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S., and Vechev, M. Ai2: Safety and robustness certification of neural networks with abstract interpretation. In *2018 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2018.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

Gowal, S., Huang, P.-S., van den Oord, A., Mann, T., and Kohli, P. Self-supervised adversarial robustness for the low-label, high-data regime. In *International Conference on Learning Representations*, 2020.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in Neural Information Processing Systems*, 33:21271–21284, 2020.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Hendrycks, D., Mazeika, M., Kadavath, S., and Song, D. Using self-supervised learning can improve model robustness and uncertainty. *arXiv preprint arXiv:1906.12340*, 2019.

Jiang, Z., Chen, T., Chen, T., and Wang, Z. Robust pre-training by adversarial contrastive learning. *Advances in Neural Information Processing Systems*, 33:16199–16210, 2020.

Jing, L. and Tian, Y. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11): 4037–4058, 2020.

Kannan, H., Kurakin, A., and Goodfellow, I. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

Katz, G., Barrett, C., Dill, D. L., Julian, K., and Kochenderfer, M. J. Reluplex: An efficient smt solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pp. 97–117. Springer, 2017.

Kim, H. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

Kim, M., Tack, J., and Hwang, S. J. Adversarial self-supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:2983–2994, 2020.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kumar, A., Levine, A., Feizi, S., and Goldstein, T. Certifying confidence via randomized smoothing. *Advances in Neural Information Processing Systems*, 33:5165–5177, 2020.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016.

Lecuyer, M., Atlidakis, V., Geambasu, R., Hsu, D., and Jana, S. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 656–672. IEEE, 2019.

Levine, A. and Feizi, S. Robustness certificates for sparse adversarial attacks by randomized ablation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 4585–4593, 2020.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C.-J. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3578–3586. PMLR, 2018.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582, 2016.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pp. 582–597. IEEE, 2016.

Raghunathan, A., Steinhardt, J., and Liang, P. S. Semidefinite relaxations for certifying robustness to adversarial examples. *Advances in Neural Information Processing Systems*, 31, 2018.

Rebuffi, S.-A., Gowal, S., Calian, D. A., Stimberg, F., Wiles, O., and Mann, T. Fixing data augmentation to improve adversarial robustness. *arXiv preprint arXiv:2103.01946*, 2021.

Salman, H., Li, J., Razenshteyn, I., Zhang, P., Zhang, H., Bubeck, S., and Yang, G. Provably robust deep learning via adversarially trained smoothed classifiers. *Advances in Neural Information Processing Systems*, 32, 2019.

Shafahi, A., Najibi, M., Ghiasi, M. A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., and Goldstein, T. Adversarial training for free! *Advances in Neural Information Processing Systems*, 32, 2019.

Teng, J., Lee, G.-H., and Yuan, Y. ℓ_1 adversarial robustness certificates: a randomized smoothing approach. 2019.

Tjeng, V., Xiao, K., and Tedrake, R. Evaluating robustness of neural networks with mixed integer programming. *arXiv preprint arXiv:1711.07356*, 2017.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018.

Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C.-J., Daniel, L., Boning, D., and Dhillon, I. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pp. 5276–5285. PMLR, 2018.

Wong, E. and Kolter, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5286–5295. PMLR, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Xu, W., Evans, D., and Qi, Y. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

Yang, G., Duan, T., Hu, J. E., Salman, H., Razenshteyn, I., and Li, J. Randomized smoothing of all shapes and sizes. In *International Conference on Machine Learning*, pp. 10693–10705. PMLR, 2020.

Zhang, D., Ye, M., Gong, C., Zhu, Z., and Liu, Q. Filling the soap bubbles: Efficient black-box adversarial certification with non-gaussian smoothing. 2019a.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. *Advances in Neural Information Processing Systems*, 32, 2019b.

A. Details of Image Transformation

We provide the full code in the supplementary materials for reproducing the results. Here we elaborate the details of image transformations used in the paper, a key part of our implementation. We utilize *random resized crop*, *random horizontal flip*, and *random color distortion* as the default image transformation. Different from the original data augmentation of SimCLR, we do not apply random Gaussian blur, as discussed in Section 4.2.

The operation constructing a pair of transformed images are detailed as below. We note that:

- $\mathcal{T}(x)$ is T_simCLR(img, flag_RdSm = False)
- $\mathcal{Q}(\mathcal{T}(x))$ is T_simCLR(img, flag_RdSm = True)

```
import torchvision
T_simCLR = TransformsSimCLR(noise_type = "uniform")
img_T_pair = torch.stack([T_simCLR(img, flag_RdSm = False), T_simCLR(img, flag_RdSm = True)])

def get_color_distortion(s=0.5):
    color_jitter = torchvision.transforms.ColorJitter(0.8*s, 0.8*s, 0.8*s, 0.2*s)
    rnd_color_jitter = torchvision.transforms.RandomApply([color_jitter], p=0.8)
    rnd_gray = torchvision.transforms.RandomGrayscale(p=0.2)
    color_distort = torchvision.transforms.Compose([rnd_color_jitter, rnd_gray])
    return color_distort

class TransformsSimCLR:
    def __init__(self, noise_type, noise_sd, size):
        self.train_transform = torchvision.transforms.Compose([
                                        torchvision.transforms.ToPILImage(),
                                        torchvision.transforms.RandomResizedCrop(size),
                                        torchvision.transforms.RandomHorizontalFlip(p=0.5),
                                        get_color_distortion(s=0.5),
                                        torchvision.transforms.ToTensor()])
        self.train_transform_randmized_smoothing = torchvision.transforms.Compose([
                                        torchvision.transforms.ToPILImage(),
                                        torchvision.transforms.RandomResizedCrop(size),
                                        torchvision.transforms.RandomHorizontalFlip(p=0.5),
                                        get_color_distortion(s=0.5),
                                        torchvision.transforms.ToTensor(),
                                        Perturb_In_Randmized_Smoothing(noise_type, noise_sd),
                                        ])
    def __call__(self, x, flag_RdSm_in_simCLR = False):
        if not flag_RdSm_in_simCLR:
            return self.train_transform(x)
        else:
            return self.train_transform_randmized_smoothing(x)


class Perturb_In_Randmized_Smoothing():
    def __init__(self, noise_type, noise_sd = 0.5):
        self.noise_type = noise_type
        self.noise_sd = noise_sd

    def __call__(self, img):
        if self.noise_type == "guassian":
            img = torch.randn_like(img)*self.noise_sd + img
        elif self.noise_type == "uniform":
            img = (torch.rand_like(img) - 0.5)*2*self.noise_sd + img
        return img
```

B. Training Setting

Recall that the proposed RUSH has a two-stage training process. (i) In the *pre-training* stage, model $g_{\theta_p} \circ f_\theta$ is trained with a SGD optimizer with batch size=512, initial learning rate=0.5, momentum=0.9, and weight decay=0.0001 for 1000 epochs. We use cosine learning rate decay during training with minimum learning rate = 0.001. The temperature parameter in contrastive loss $\mathcal{L}_{pre}$ is set to 0.5. (ii) In the *fine-tuning* stage, model $\phi_{\theta_c} \circ f_\theta$ is trained with the frozen feature extractor f_θ. We note that as we believe batch normalization should be based on specific feature distributions, so the bias of the BatchNorm layers of f_θ is not frozen during fine-tuning. We use a SGD optimizer with batch size=512, initial learning rate=0.1, momentum=0.9, and weight decay=0.0002 to train the network for 25 epochs. We use cosine learning rate decay during training with minimum learning rate = 0.001.

C. Adversarial Perturbation under Randomized Smoothing and Inference

Figure 1. Adversarial perturbation under randomized smoothing