RETHINKING THE EFFECT OF DATA AUGMENTATION IN ADVERSARIAL CONTRASTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent works have shown that self-supervised learning can achieve remarkable robustness when integrated with adversarial training (AT). However, the robustness gap between supervised AT (sup-AT) and self-supervised AT (self-AT) remains significant. Motivated by this observation, we revisit existing self-AT and discover an inherent dilemma that affects self-AT robustness: either strong or weak data augmentations are harmful to self-AT, and a medium strength is insufficient to bridge the gap. To resolve this dilemma, we propose a simple remedy named DynACL (Dynamic Adversarial Contrastive Learning). In particular, we propose an augmentation schedule that gradually anneals from a strong augmentation to a weak one to benefit from both extreme cases. Besides, we adopt a fast post-processing stage for adapting it to downstream tasks. Through extensive experiments, we show that DynACL can improve the state-of-the-art self-AT robustness by 8.84% under Auto-Attack on the CIFAR-10 dataset, and can even outperform vanilla supervised adversarial training. We demonstrate that self-supervised AT can attain even better robustness than supervised AT for the first time.

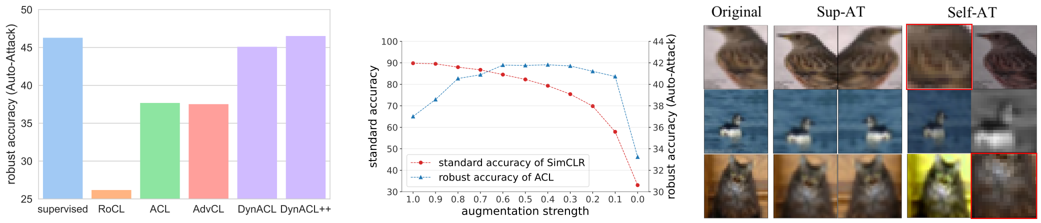
1 INTRODUCTION

Learning low-dimensional representations of inputs without supervision is one of the ultimate goals of machine learning. As a promising approach, self-supervised learning is rapidly closing the performance gap with respect to supervised learning (He et al., 2016; Chen et al., 2020b) in downstream tasks. However, for whatever supervised and self-supervised learning models, adversarial vulnerability remains a widely-concerned security issue, *i.e.,* natural inputs injected by small and human imperceptible adversarial perturbations can fool the deep neural networks (DNNs) into making wrong predictions.

For supervised learning, one popular approach to enhance adversarial robustness is adversarial training (**sup-AT**), where DNNs are trained on adversarial examples (Goodfellow et al., 2014; Madry et al., 2017). However, sup-AT requires true labels to craft adversarial examples. For self-supervised learning, recent works including RoCL (Kim et al., 2020), ACL (Jiang et al., 2020), and AdvCL (Fan et al., 2021) have proposed the corresponding adversarial training counterpart (**self-AT**). Despite obtaining a certain degree of robustness, these methods failed to eliminate the large gap between sup-AT and self-AT. As a reference, for standard training (ST), the performance gap between sup-ST and self-ST is lower than 1% on CIFAR-10 (da Costa et al., 2022). However, for adversarial contrastive training (AT), the robustness gap remains $> 8\%$ (sup-AT 46.2% v.s. self-AT 37.6%), as shown in Figure 1(a). This phenomenon leads us to the following question:

What is the key factor that prevents self-AT from obtaining comparable robustness to sup-AT?

To answer this question, we examine the difference between sup-AT and self-AT. As they share the same perturbation strategy, the main difference mainly lies in their learning paradigms. Different from sup-AT that directly utilizes labels, existing self-AT methods (RoCL, ACL, and AdvCL) all adopt the contrastive learning paradigm, which instead relies crucially on strong data augmentations to obtain generalizable representations. As noted by recent theoretical understandings of contrastive learning (Wang et al., 2022; HaoChen et al., 2021), strong augmentations create support overlap between intra-class samples such that the alignment between augmented pairs could cluster these intra-class samples together. Under weaker augmentations, the accuracy of contrastive learning will dramatically decrease, as the red line shows in Figure 1(b). Nevertheless, the blue line in Figure 1(b) demonstrates that strong data augmentations are very harmful to adversarial robustness. This reveals



(a) robustness comparison (b) standard contrastive learning (c) data augmentations

Figure 1: Experiments on CIFAR-10: (a) Comparison of supervised AT (vanilla PGD-AT (Madry et al., 2017)) and five self-supervised AT methods: RoCL, ACL, AdvCL, our DynACL and DynACL++ with ResNet-18 backbone. (b) Performance of standard contrastive learning (Chen et al., 2020a) using different augmentation strengths. (c) Illustrative examples of data augmentations adopted by sup-AT and self-AT. We can see that self-AT adopts much more aggressive augmentation than sup-AT.

a critical dilemma of self-AT, *i.e.,* the default equipped strong augmentations, although useful for standard accuracy, will hurt adversarial robustness. Further, we dive into the strong augmentations adopted by self-AT in Figure 1(c) and find that significant semantic shifts to the original images are observed. Therefore, there will be a much larger distribution gap between training and test data, and the local ℓ_p robustness on training examples becomes less transferable to test data. More severely, some augmented samples from different classes become even inseparable (the bird and the cat samples in red squares), and thus largely distort the decision boundary. The necessity and harmfulness of data augmentations form a fundamental dilemma in self-AT.

Therefore, as data augmentations play an essential role in self-AT, we need to strike a balance between utilizing strong augmentations for representation learning and avoiding large image distortions for good robustness. Although it seems impossible under the current static augmentation strategy, we notice that we can alleviate this dilemma by adopting a *dynamic* augmentation schedule. In particular, we could learn good representations with aggressive augmentations at the beginning stage, and gradually transfer the training robustness to the test data by adopting milder and milder augmentations at later stages. We name this method as Dynamic Adversarial Contrastive Learning (DynACL). In this way, DynACL could benefit from both sides, gaining both representation power and robustness aligned with test distribution. Built upon DynACL, we further design a fast post-processing stage for bridging the difference between the pretraining and downstream tasks, dubbed as DynACL++. As a preview of results, Figure 1(a) shows that the proposed DynACL and DynACL++ bring a significant improvement over state-of-the-art self-AT methods, and achieve comparable, or even superior robustness, to vanilla sup-AT. Our main contributions are:

- We reveal the reason behind the robustness gap between self-AT and sup-AT, *i.e.,* the widely adopted aggressive data augmentations in self-supervised learning may bring the issues of training-test distribution shift and class inseparability.
- We propose a dynamic augmentation strategy along the training process to balance the need of strong augmentations for representation and mild augmentations for robustness, called Dynamic Adversarial Contrastive Learning (DynACL).
- Experiments show that our proposed methods improve both clean accuracy and robustness over existing self-AT methods by a large margin. Notably, DynACL++ improves the robustness of ACL (Jiang et al., 2020) from 37.62% to 46.46% on CIFAR-10, which is even slightly better than vanilla supervised AT (Madry et al., 2017). Meanwhile, it is also more computationally efficient as it requires less training time than ACL.

2 BACKGROUND AND RELATED WORK

Sup-AT. Given a labeled training dataset $\mathcal{D}_l = \{(\bar{x}, y) | \bar{x} \in \mathbb{R}^n, y \in [K]\}$, to be resistant to ℓ_p-bounded adversarial attack, supervised adversarial training (sup-AT) generally adopts the following min-max framework (Madry et al., 2017):

$$\mathcal{L}_{\text{sup}-\text{AT}}(f) = \mathbb{E}_{\bar{x},y} \max_{\delta_{\bar{x}} \in \Delta} \ell_{\text{CE}}(f(\bar{x} + \delta_{\bar{x}}), y), \quad \ell_{\text{CE}}(f(\cdot), y) = -\log \frac{\exp(f(\cdot)[y])}{\sum_{k=1}^{K} \exp(f(\cdot)[k])}. \tag{1}$$

Here the classifier $f : \mathbb{R}^n \to \mathbb{R}^K$ is learned on the adversarially perturbed data $(\bar{x} + \delta_{\bar{x}}, y)$, and $\Delta = \{\delta : \|\delta\|_p \leq \varepsilon\}$ denotes the feasible set for adversrial perturbation.

Contrastive Learning. For each $\bar{x} \in \mathcal{X}$, we draw two positive samples x, x^+ from the augmentation distribution $A(\cdot|\bar{x})$, and draw M independent negative samples $\{x_m^-\}_{m=1}^M$ from the marginal distribution $A(\cdot) = \mathbb{E}_{\bar{x}} A(\cdot|\bar{x})$. Then, we train an encoder $g : \mathcal{X} \to \mathcal{Z}$ by the widely adopted InfoNCE loss using the augmented data pair $(x, x^+, \{x_m^-\})$ (Oord et al., 2018):

$$\begin{aligned} \min_g \mathcal{L}_{\mathrm{NCE}}(g) = \; & \mathbb{E}_{x,x^+,\{x_m^-\}} \ell_{\mathrm{NCE}}(x, x^+, \{x_m^-\}; g), \\ \ell_{\mathrm{NCE}}(x, x^+, \{x_m^-\}; g) = & -\log \frac{\exp(\mathrm{sim}(g(x), g(x^+))/\tau)}{\sum_m \exp(\mathrm{sim}(g(x), g(x_m^-))/\tau)}. \end{aligned} \tag{2}$$

Here $\mathrm{sim}(\cdot, \cdot)$ is the cosine similarity between two vectors, and τ is a temperature hyperparameter. Surged from InfoNCE (Oord et al., 2018), contrastive learning undergoes a rapid growth (Tian et al., 2019; Misra & Maaten, 2020; Chen et al., 2020a; He et al., 2020) and has demonstrated state-of-the-art performance on self-supervised tasks (Chen et al., 2020b; Kong et al., 2020). Nevertheless, several works also point out contrastive learning is still vulnerable to adversarial attack when we transfer the learned features to the downstream classification (Ho & Nvasconcelos, 2020; Kim et al., 2020).

Self-AT. To enhance the robustness of contrastive learning, adversarial training was similarly adapted to self-supervised settings (self-AT). Since there are no available labels, adversarial examples are generated by maximizing the contrastive loss (Eq. 2) *w.r.t.* all input samples,

$$\ell_{\mathrm{AdvNCE}}(x, x^+, \{x_m^-\}; g) = \max_{\delta, \delta^+, \{\delta_m^-\} \in \Delta} \ell_{\mathrm{NCE}}(x + \delta, x^+ + \delta^+, \{x_m^- + \delta_m^-\}; g), \tag{3}$$

Several previous works, including ACL (Jiang et al., 2020), RoCL (Kim et al., 2020), and CLAE (Ho & Nvasconcelos, 2020), adopt Eq. 3 to perform self-AT. In addition, ACL (Jiang et al., 2020) further incorporates the dual-BN technique (Xie et al., 2020) for better performance. Specifically, given a backbone model like ResNet, they duplicate its BN modules and regard the encoder g as two different branches (all parameters are shared except for BN): the clean branch g_c and the adversarial branch g_a are trained by clean examples and adversarial examples respectively:

$$\ell_{\mathrm{ACL}}(x, x^+, \{x_m^-\}; g) = \ell_{\mathrm{NCE}}(x, x^+, \{x_m^-\}; g_c) + \ell_{\mathrm{AdvNCE}}(x, x^+, \{x_m^-\}; g_a). \tag{4}$$

After training, they use the adversarial branch g_a as the robust encoder. They empirically show that ACL obtains better robustness than the single branch version. Built upon ACL, AdvCL (Fan et al., 2021) further leverages an additional ImageNet-pretrained model to generate pseudo-labels for sup-AT via clustering. Different from them, we explore an orthogonal direction to previous work by investigating the influence of augmentation strength on the robustness of self-AT.

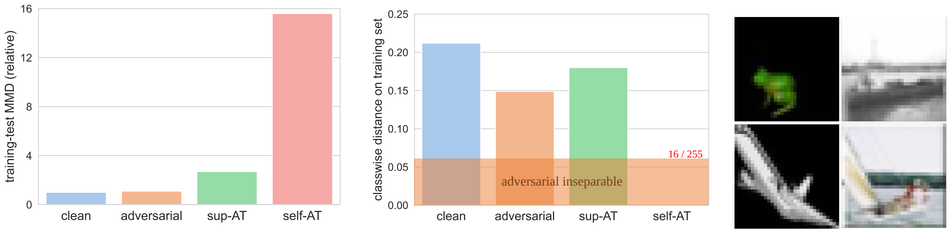
3 RETHINKING THE EFFECT OF DATA AUGMENTATIONS IN SELF-AT

Different from the case of supervised learning, data augmentation is an *indispensable* ingredient in contrastive self-supervised learning. Existing self-AT methods inherit the aggressive augmentation strategy adopted in standard contrastive learning (*e.g.,* SimCLR (Chen et al., 2020a)), *i.e.,* a composition of augmentations like random resized crop, color jitter, and grayscale. However, this strategy is *not* necessarily optimal for adversarial training. In this section, we provide an in-depth investigation of the effect of this default data augmentation strategy on self-supervised adversarial training (self-AT).

3.1 PROBLEMS OF AGGRESSIVE DATA AUGMENTATIONS IN SELF-AT

As shown in Figure 1(c), the default augmentations in self-AT are much more aggressive than that of sup-AT, which leads to two obvious drawbacks: large training-test distribution gap and class inseparability, as we quantitatively characterize below. For comparison, we evaluate 4 kinds of data augmentation methods on CIFAR-10: 1) clean (no augmentation), 2) pre-generated adversarial augmentation by a pretrained classifier, 3) the default augmentations (padding-crop and horizontal flip) adopted by sup-AT (Madry et al., 2017), and 4) the default augmentations (random resized crop, color jitter, grayscale, and horizontal flip) adopted by self-AT (Jiang et al., 2020; Kim et al., 2020).

Large Training-test Distribution Gap. Although training and test samples are supposed to follow the same distribution, random augmentations are often adopted in the training stage, which creates a



(a) training-test distribution gap (b) minimal classwise distance (c) selected samples

Figure 2: Comparison of different augmentation strategies on CIFAR-10, where "clean" denotes no augmentation; "adversarial" denotes adversarial examples generated by PGD attack; "sup-AT" and "self-AT" denote the corresponding default data augmentations.

training-test distribution gap. To study it quantitatively in practice, we measure this distribution gap with Maximum Mean Discrepancy (MMD) (Gretton et al., 2012) calculated between the augmented training set and raw test set of CIFAR-10 (details in Appendix B.1).[1] As shown in Figure 2(a), "adversarial" and "sup-AT" augmentations only produce a small distribution gap, while the aggressive "self-AT" augmentations yield a much larger gap (6 times) than "sup-AT". Therefore, the self-AT training data could be very distinct from test data, and the robustness obtained *w.r.t.* augmented training data could be severely degraded when transferred to the test data.

Class Inseparability. Yang et al. (2020) found that under supervised augmentations and adversarial perturbations with $\varepsilon = 8/255$, the minimal ℓ_∞ distance between training samples from different classes, namely the *classwise distance*, is larger than $2 \cdot \varepsilon = 16/255$. In other words, CIFAR-10 data are *adversarially separable* in sup-AT. However, we notice that this *no longer* holds in self-AT. As shown in Figure 2(b), self-AT augmentations have a much smaller classwise distance that is close to zero. One possible cause is those CIFAR-10 samples with complete black or white regions, which could result in identical augmentations after aggressive random cropping (Figure 2(c)). As a result of class inseparability, self-AT will mix adversarial examples from different classes together, and these noisy examples will degrade the robustness of the learned decision boundary.

3.2 QUANTIFYING THE EFFECT OF DIFFERENT AUGMENTATION STRENGTHS

The discussion above shows that aggressive augmentations are harmful for self-AT, which motivates us to quantitatively study the effect of different augmentation strengths on self-AT.

Characterization of Augmentation Strength. To begin with, we need to define a quantitative measure of the augmentation strength. To achieve this, we design a set of data augmentations $\mathcal{T}(s)$ that vary according to a hyperparameter $s \in [0, 1]$: 1) when $s = 0$, it is mild and contains only horizontal flip; 2) when $s = 1$, it is the aggressive augmentations of self-AT; and 3) when $s \in (0, 1)$, we linearly interpolate the augmentation hyperparameters to create a middle degree of augmentation. For example, in random resized cropping, a larger s will crop to a smaller region in an image, which amounts to a larger training-test distribution gap, as shown in Figure 3. A detailed configuration of $\mathcal{T}(s)$ is listed in Appendix A.1.

A Static Augmentation Strength Cannot Fully Resolve the Dilemma of Self-AT. The discussion in Section 3.1 reveals that strong augmentations are harmful for adversarial training while being critical for representation learning (Figure 1(b)). The augmentation strength thus becomes a critical dilemma of self-AT. Here, we study whether we could resolve this issue by tuning the augmentation strength s for an optimal tradeoff. As shown in Figures 4(a) and 4(b), decreasing augmentation strength s (left to right) indeed brings smaller training-test distribution gap and eliminates class inseparability, which benefits adversarial robustness (blue line in Figure 4(c)) and hurts standard accuracy at the same time (red line in Figure 4(c)). Roughly, $s = 0.5$ seems a good trade-off between accuracy and robustness, while its 41.79% robustness is still much lower than 46.23% robustness of sup-AT (orange line in Figure 4(c)). This suggests that the dilemma on augmentation strength cannot be fully resolved by tuning a static augmentation strength, although useful to a certain degree. The training process in Figure 4(d) presents that strong or weak augmentations have their own deflects while medium augmentations only bring a very limited gain (green line). In the next section, we will introduce a

---

[1] As adversarial perturbations are defined in the input space, we measure the distance in the input space to evaluate adversarial vulnerability. We include latent-space results in Appendix B.2, which show similar trends.

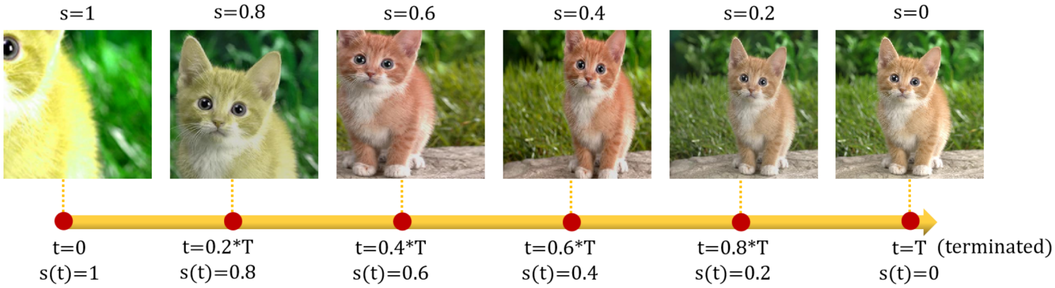
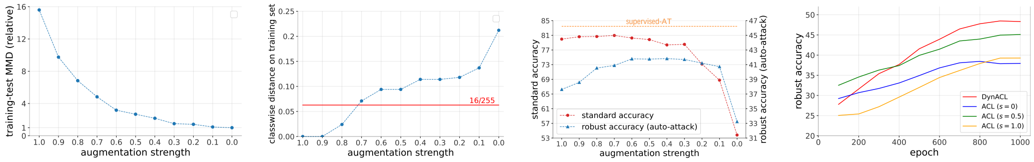


Figure 3: Illustration of augmenting the same image of cat with different augmentation strengths $s \in [0, 1]$, where a smaller s (left to right) indicates milder image distortion.



(a) training-test distribution gap (b) minimal classwise distance (c) accuracy and robustness (d) training process

Figure 4: Quantitative results on CIFAR-10 of self-AT (baseline method is ACL (Jiang et al., 2020)) with different augmentation strengths.

new dynamic augmentation strategy to resolve this dilemma (a preview of performance is shown as red line in Figure 4(d)).

4 PROPOSED DYNAMIC ADVERSARIAL CONTRASTIVE LEARNING (DYNACL)

In this section, we propose a simple but effective method **DynACL** to resolve the dilemma on self-AT augmentations. In Section 4.1, we introduce our dynamic augmentation strategy for self-AT methods and propose DynACL. In Section 4.2, we further propose an improved version of DynACL with a fast post-processing stage to mitigate pretraining and downstream tasks, named DynACL++. Figure 6 in Appendix A.3 demonstrates the framework of the proposed DynACL.

4.1 DYNACL: DYNAMIC ADVERSARIAL CONTRASTIVE LEARNING

Dynamic Augmentation. Different from prior works that all adopt a static augmentation strength, we propose to treat the augmentation strength s as a dynamic parameter $s(t)$ that varies along different epochs t. In this way, at earlier training stages, our self-AT can benefit from aggressive augmentations to learn meaningful representations. As training progresses, augmentation strength s decays from 1 to 0, the training-test distribution gap will be gradually closed and the classwise distance grows larger and separable (both can be seen from Figures 4(a) and 4(b)).

Following this idea, we adopt the following piecewise decay augmentation schedule:

$$s(t) = 1 - \lfloor \frac{t}{K} \rfloor \cdot \frac{K}{T}, \quad t = 0, \ldots, T-1, \tag{5}$$

where T is the total number of training epochs, K is the decay period, and $\lfloor \cdot \rfloor$ is the floor operation. As illustrated in Figure 3, $s(t)$ starts from the standard augmentation strength at the start epoch $s(0) = 1$, and decays by reducing K/T every K epochs. Meanwhile, we also reweight the natural and adversarial InfoNCE loss according to the dynamic augmentation strength $s(t)$. To be precise, for pretraining at t-th epoch, we adopt the following DynACL (ANnealing ACL) loss:

$$\mathcal{L}_{\text{DynACL}}(g; t) = (1 - w(t))\mathcal{L}_{\text{NCE}}(g_c; s(t)) + (1 + w(t))\mathcal{L}_{\text{AdvNCE}}(g_a; s(t)). \tag{6}$$

Here we assign the adjusted weight $w(t)$ according to the augmentation strength $s(t)$ by

$$w(t) = \lambda(1 - s(t)), \ \ t = 0, \ldots, T-1, \tag{7}$$

where λ denotes the reweighting rate. Contrary to augmentation strength $s(t)$, the weight $w(t)$ rises from 0 to λ as training proceeds. In this way, we gradually upweight the adversarial InfoNCE loss rather than adopting a fixed interpolation ratio. This is because at earlier training stages, the natural loss is beneficial for learning useful features from unlabeled data. Afterward, the adversarial loss $\mathcal{L}_{\mathrm{AdvNCE}}$ contributes more to model robustness, so we adopt a larger weight $w(t)$ at later stages as $t \to T-1$.

4.2 DYNACL++: DYNACL WITH FAST POST-PROCESSING

The main dynamic pretraining phase proposed above can help mitigate the training-test data distribution gap. Nevertheless, there remains a training-test task discrepancy, which lies in the difference between the contrastive pretraining and downstream classification tasks: the former is an *instance-level* discriminative task, while the latter is a *class-level* discriminative task. Therefore, the pretrained instance-wise robustness might not transfer perfectly to the classwise robustness to be evaluated at test time.

Different from the heavy engineering of the pretraining phase as in AdvCL (Fan et al., 2021), we propose a simple post-processing phase with pseudo adversarial training (PAT) to mitigate this "task gap". Given a pretrained encoder $g=(g_c,g_a)$ (Eq. 6) consisting of a clean branch g_c and an adversarial branch g_a, we simply generate pseudo labels $\{\hat{y}_i\}$ with k-means clustering based on the clean branch g_c, whose natural accuracy is relatively high because it is fed with natural samples during training,

$$\text{(clustering)}\ \ \{\hat{y}_i\} = \mathrm{k\_means}(\{x_i\}; g_c). \tag{8}$$

Afterward, we apply a linear head $h:\mathbb{R}^m \to \mathbb{R}^k$ (k is the number of classes) on top of the adversarial branch g_a, and adversarially finetune the whole network $h \circ g_a$ on pseudo pairs $(x, \hat{y})$. To be specific, we first freeze the encoder and align the classification head by standard training. Then we adopt TRADES (Zhang et al., 2019) and finetune the whole network. The whole process is named as pseudo adversarial training (PAT):

$$\text{(pseudo adversarial training)}\ \mathcal{L}_{\mathrm{PAT}}(h \circ g_a) = \mathbb{E}_{\bar{x},y}\left\{\ell_{\mathrm{CE}}(f(\bar{x}),y) + \max_{\delta_{\bar{x}}\in\Delta} \mathrm{KL}(f(\bar{x})||f(\bar{x}+\delta_{\bar{x}}))\right\}. \tag{9}$$

Note that the augmentation strength $s(t)$ has already been reduced to 0 at the end of the pretraining stage, so we only use mild augmentations for PAT as in canonical sup-AT (Madry et al., 2017). In practice, we only perform PAT for a few epochs, *e.g.,* $T'=10$ for the first phase and $T''=25$ for the second phase, which is much shorter than the pretraining stage with typically 1000 epochs. Thus, we regard this as a fast post-processing stage of the learned features to match the downstream classification task. The overall algorithm is presented in Appendix A.2.

4.3 COMPARISON TO PREVIOUS WORKS

Difference to ACL. Our DynACL is built upon the dual-stream framework of ACL (Jiang et al., 2020), while differing from ACL in three aspects. Firstly, DynACL adopts a dynamic augmentation schedule to bridge the training-test distribution gap via annealing, while ACL adopts a constant augmentation policy. Secondly, DynACL uses SimSiam as the backbone instead of SimCLR. Lastly, ACL simply drops the natural encoder g_c after pretraining, while DynACL++ utilizes it to generate pseudo labels for pseudo adversarial training. In Section 5.2, we show that each innovation brings significant improvements in robustness.

Difference to AdvCL. Also built upon ACL, AdvCL (Fan et al., 2021) incorporates an additional high-frequency view and clustering-based pseudo labels into ACL during *the entire pretraining phase*. Thus, AdvCL spends two times training time than ACL while DynACL requires less training time than ACL. Additionally, AdvCL generates pseudo labels by an ImageNet-pretrained model, while DynACL++ generates pseudo labels by the pretrained model itself. Lastly, even with ImageNet labels, AdvCL still performs worse than DynACL (Section 5.1).

5 EXPERIMENTS

In this section, we evaluate DynACL and DynACL++ under the benchmark datasets: CIFAR-10, CIFAR-100 (Krizhevsky, 2009), and STL-10 (Coates et al., 2011), compared with baseline methods:

Table 1: Comparison of supervised and self-supervised adversarial training methods on CIFAR-10, CIFAR-100, and STL-10. SA and AA stand for standard accuracy and robust accuracy under Auto-Attack (Croce & Hein, 2020). AdvCL (+ImageNet) uses additional ImageNet data. N/A: AdvCL does not provide ImageNet labels for STL-10.

Pretraining Method	CIFAR-10		CIFAR-100		STL-10	
	AA(%)	SA(%)	AA(%)	SA(%)	AA(%)	SA(%)
Sup-AT	46.23	84.35	23.27	58.98	29.21	49.38
RoCL	26.12	77.90	8.72	42.93	26.51	**78.19**
ACL	37.62	79.32	15.68	45.34	33.24	71.21
AdvCL	37.46	73.23	15.45	37.58	45.26	72.11
DynACL (ours)	45.04	77.41	19.25	45.73	46.59	69.67
DynACL++ (ours)	**46.46**	**79.81**	**20.05**	**52.26**	**47.21**	70.93
AdvCL (+ImageNet)	42.57	80.85	19.78	48.34	N/A	N/A

RoCL (Kim et al., 2020), ACL (Jiang et al., 2020), and AdvCL (Fan et al., 2021). We briefly introduce the basic experimental setup in the following, and more experimental details can be found in Appendix C.

Pretraining. We adopt ResNet-18 (He et al., 2016) as the encoder following existing self-AT methods (Kim et al., 2020; Jiang et al., 2020; Fan et al., 2021), and inherit all training configurations of ACL (Jiang et al., 2020) except that we adopt SimSiam Chen & He (2020) as the backbone instead of SimCLR Chen et al. (2020a). We set the decay period $K = 50$, and reweighting rate $\lambda = 2/3$.

Evaluation Protocols. We evaluate the learned representations with three protocols: standard linear finetuning (SLF), adversarial linear finetuning (ALF), and adversarial full finetuning (AFF). The former two protocols freeze the learned encoder and tune the linear classifier using cross entropy loss with natural (SLF) or adversarial (ALF) samples, respectively. As for AFF, we adopt the pretrained encoder as weight initialization and train the whole classifier following ACL (Jiang et al., 2020).

5.1 BENCHMARKING THE PERFORMANCE OF DYNACL

Robustness on Various Datasets. In Table 1, we evaluate the robustness of sup-AT and self-AT methods on CIFAR-10, CIFAR-100, and STL-10. As we can see, DynACL outperforms all previous self-AT methods in terms of robustness without using additional data[2]. Specifically, under the auto-attack benchmark, DynACL brings a big leap of robustness by improving previous state-of-the-art self-AT methods by 8.84% (from 37.62% to 46.46%) on CIFAR-10, 4.37% (from 15.68% to 20.05%) on CIFAR-100, and 1.95% on STL-10 (from 45.26% to 47.21%). Furthermore, on CIFAR-10, DynACL++ achieves even higher AA accuracy than the supervised vanilla AT (with heavily tuned default hyperparameters (Pang et al., 2020)). It is the first time that a self-supervised AT method outperforms its supervised counterpart.

Robustness under Different Evaluation Protocols. Table 2 shows that DynACL and DynACL++ obtain state-of-the-art robustness among self-AT methods across different evaluation protocols. In particular, under ALF settings, DynACL++ outperforms all existing self-AT methods and even sup-AT method. Furthermore, under AFF settings, DynACL could improve the heavily tuned sup-AT baseline (Pang et al., 2020) by 1.58% AA accuracy ($48.96\% \rightarrow 50.54\%$), and is superior to other pretraining methods. In conclusion, DynACL and DynACL++ have achieved state-of-the-art performance across various evaluation protocols.

Performance under Semi-supervised Settings Following Jiang et al. (2020), we evaluate our proposed DynACL++ under semi-supervised settings. Baseline methods are state-of-the-art semi-supervised AT method UAT++ [3] (Uesato et al., 2019) and self-supervised based method ACL. We use the same evaluation protocol in (Jiang et al., 2020), and the results are shown in Table 3. We

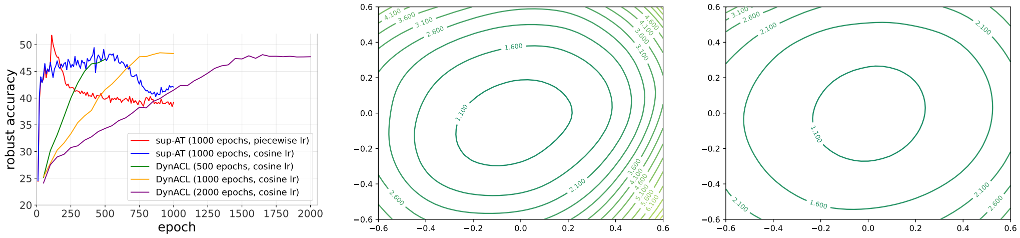
---

[2] Note that AdvCL incorporates extra data because it utilizes an ImageNet-pretrained model to generate pseudo labels. For a fair comparison, we replace AdvCL's ImageNet model with a model pretrained on the training dataset itself. We use AdvCL to denote this ablated version and refer to the original version as "AdvCL (+ImageNet)".

[3] UAT++ (Uesato et al., 2019) did not release the training code or pretrained ResNet-18 model. We copied the RA and SA results from ACL (Jiang et al., 2020).

Table 2: Performance comparison on CIFAR-10 with three evaluation protocols: SLF (standard linear finetuning), ALF (adversarial linear finetuning), and AFF (adversarial full finetuning).

Pretraining Method	SLF		ALF		AFF	
	AA(%)	SA(%)	AA(%)	SA(%)	AA(%)	SA(%)
Sup-AT	46.23	84.35	47.00	83.22	48.96	80.23
RoCL	26.12	77.90	29.69	75.62	45.02	78.51
ACL	37.62	79.32	40.91	76.57	49.46	82.11
AdvCL	37.46	73.23	37.28	73.15	48.58	**82.31**
DynACL (ours)	45.04	77.41	45.87	72.87	**50.54**	81.84
DynACL++ (ours)	**46.46**	**79.81**	**47.95**	**78.84**	50.31	81.94
AdvCL (+ImageNet)	42.41	81.25	42.54	79.41	49.97	83.27



(a) robustness v.s. training progress (b) loss landscape of ACL (c) loss landscape of DynACL++

Figure 5: (a) Comparison of Robust Overfitting between sup-AT and self-AT (DynACL) with different pretraining schedules (without post-processing) on CIFAR-10. (b), (c): loss landscape visualization of ACL and DynACL++. Note that DynACL++ enjoys a flatter loss landscape compared with ACL.

observe that our DynACL++ can demonstrate fairly good performance even when only 1% of labels are available.

Table 3: Performance under semi-supervised settings.

Label Ratio	UAT++			ACL			**DynACL++ (ours)**		
	AA(%)	RA(%)	SA(%)	AA(%)	RA(%)	SA(%)	AA(%)	RA(%)	SA(%)
1% labels	N/A	30.46	41.88	45.65	50.46	74.76	**46.95**	**51.30**	**76.77**
10% labels	N/A	50.43	70.79	45.47	50.01	75.14	**48.56**	**53.00**	**78.34**

5.2 EMPIRICAL UNDERSTANDINGS

In this part, we conduct comprehensive experiments on CIFAR-10 to have a deep understanding of the proposed DynACL and DynACL++. More analysis results are deferred to Appendix D.

Ablation Studies. We conduct an ablation study of our two major contributions (augmentation annealing in Section 4.1 and post-processing in Section 4.2) in Table 4. We can see that both annealing and post-processing can significantly boost model's robustness, which demonstrates the superiority of our method.

Robust Overfitting. A well-known pitfall of sup-AT is *Robust Overfitting* (Rice et al., 2020): when training for longer epochs, the test robustness will dramatically degrade. Surprisingly, we find that this phenomenon does not exist in DynACL. In Figure 5(a), we compare DynACL and sup-AT for very long (pre)training epochs. We can see that sup-AT overfits quickly after the 100-th epoch and suffers a 12% test robustness decrease eventually. Nevertheless, the test robustness of DynACL keeps increasing *along the training process* even under 2000 epochs.

Loss Landscape. Previous studies (Prabhu et al., 2019; Yu et al., 2018; Wu et al., 2020) have shown that a flatter weight landscape often leads to better robust generalization. Motivated by that discovery,

Table 4: Ablation study two key designs of our method. Baseline method is ACL (Jiang et al., 2020), and last two lines represent DynACL and DynACL++, respectively.

Annealing	Post-processing	AA(%)	RA(%)	SA(%)
×	×	37.62	40.44	79.32
×	✓	43.24 **(+5.62)**	45.48 **(+5.04)**	77.40 **(-2.92)**
✓	×	45.04 **(+7.42)**	48.40 **(+7.96)**	77.41 **(-2.91)**
✓	✓	46.46 **(+8.47)**	49.21 **(+8.49)**	79.81 **(+0.49)**

Table 5: Performance of DynACL++ on CIFAR-10 with alternative annealing schedules and decay periods K (in Eq. 5) (a), and with different reweighting rate λ (in Eq. 7) (b).

(a) decay schedule and decay period K

schedule	K	AA(%)	SA(%)
constant	-	40.76	70.83
linear	1	45.77	**80.26**
	25	45.99	79.62
	50	**46.46**	79.81
	100	45.79	79.79

(b) reweighting rate λ

λ	AA(%)	SA(%)
0	44.81	77.83
1/3	45.63	79.55
1/2	46.23	79.85
2/3	**46.46**	79.81
1	46.01	**80.46**

we visualize the loss landscape of ACL and DynACL++ following (Li et al., 2018). Figure 5(b) and 5(c) show that DynACL++ has a much flatter landscape than ACL, explaining the superiority of DynACL++ in robust generalization.

Training Speed. We evaluate the total training time of self-AT methods on the same device with a single RTX 3090 GPU. Specifically, the total training time is 32.7, 105.0, 29.4, and 30.3 hours for ACL, AdvCL, DynACL, and DynACL++, respectively. Built upon ACL, our DynACL spends less training time. Moreover, since post-processing only lasts few epochs, DynACL++ brings negligible computational overhead. In comparison, AdvCL triples the training time compared with ACL. These statistics show that our DynACL and DynACL++ are both efficient and effective.

Effect of Annealing Schedule. We evaluate DynACL++ with different decay period K in Table 5(a). We observe that our dynamic schedule can significantly outperform the constant one. Also, we notice that step-wise annealing strategy performs slightly better. This may because over-frequent data reloading (i.e. over-frequent distribution shift) may make the model fail to generalize well under each distribution.

Effect of Reweighting Rate. We also study the effect of annealing loss (Eq. 7) with different reweighting rates λ. Table 5(b) shows that compared to no annealing ($\lambda = 0$), the annealing loss ($\lambda > 0$) consistently improves both accuracy and robustness for around 1%, and the best robustness is obtained with a moderate reweighting rate $\lambda = 2/3$.

6 CONCLUSIONS

In this paper, we observe a dilemma about the augmentation strength that either weak or strong augmentations degrade the model robustness through a quantitative investigation of data augmentations in self-supervised adversarial training (self-AT). Going beyond the simple trade-off by selecting a medium constant augmentation strength, we propose to adopt a dynamic augmentation schedule that gradually anneals the strength from strong to weak, named DynACL. Afterward, we further devise a fast clustering-based post-processing technique to adapt the model for downstream finetuning, named DynACL++. Experiments show that the proposed DynACL and DynACL++ successfully boost self-AT's robustness and even outperform its vanilla supervised counterpart on CIFAR-10 under Auto-Attack.

REFERENCES

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. *arXiv preprint arXiv:2011.10566*, 2020.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15, pp. 215–223, 2011.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Victor Guilherme Turrisi da Costa, Enrico Fini, Moin Nabi, Nicu Sebe, and Elisa Ricci. solo-learn: A library of self-supervised methods for visual representation learning. *Journal of Machine Learning Research*, 23(56):1–6, 2022.

Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? In *NeruIPS*, 2021.

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

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773, 2012.

Jeff Z HaoChen, Colin Wei, Adrien Gaidon, and Tengyu Ma. Provable guarantees for self-supervised deep learning with spectral contrastive loss. In *NeurIPS*, 2021.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020.

Chih-Hui Ho and Nuno Nvasconcelos. Contrastive learning with adversarial examples. In *NeruIPS*, 2020.

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

Minseon Kim, Jihoon Tack, and Sung Ju Hwang. Adversarial self-supervised contrastive learning. In *NeurIPS*, 2020.

Lingpeng Kong, Cyprien de Masson d'Autume, Wang Ling, Lei Yu, Zihang Dai, and Dani Yogatama. A mutual information maximization perspective of language representation learning. In *ICLR*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. 2009.

Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.

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.

Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. *arXiv preprint arXiv:2010.00467*, 2020.

Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and John Whaley. Understanding adversarial robustness through loss landscape geometries. *arXiv preprint arXiv:1907.09061*, 2019.

Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In *ICML*, 2020.

Aman Sinha, Hongseok Namkoong, Riccardo Volpi, and John Duchi. Certifying some distributional robustness with principled adversarial training. *arXiv preprint arXiv:1710.10571*, 2017.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

Jonathan Uesato, Jean-Baptiste Alayrac, Po-Sen Huang, Robert Stanforth, Alhussein Fawzi, and Pushmeet Kohli. Are labels required for improving adversarial robustness? In *NeurIPS*, 2019.

Yifei Wang, Qi Zhang, Yisen Wang, Jiansheng Yang, and Zhouchen Lin. Chaos is a ladder: A new theoretical understanding of contrastive learning via augmentation overlap. *arXiv preprint arXiv:2203.13457*, 2022.

Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020.

Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L Yuille, and Quoc V Le. Adversarial examples improve image recognition. In *CVPR*, 2020.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In *NeurIPS*, 2020.

Fuxun Yu, Chenchen Liu, Yanzhi Wang, Liang Zhao, and Xiang Chen. Interpreting adversarial robustness: A view from decision surface in input space. *arXiv preprint arXiv:1810.00144*, 2018.

Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

A ALGORITHM DETAILS

A.1 DETAILED CONFIGURATION OF DATA AUGMENTATIONS $\mathcal{T}(s)$

We formulate the augmentation configuration $\mathcal{T}(s)$ with respect to augmentation strength s in the following PyTorch-style code:

```python
from torchvision import transforms

def get_transforms(strength):
    rnd_color_jitter = transforms.RandomApply([transforms.ColorJitter(
            0.4 * strength, 0.4 * strength, 0.4 * strength, 0.1 *
    strength)], p=0.8 * strength)
    rnd_gray = transforms.RandomGrayscale(p=0.2 * strength)
    transform = transforms.Compose([
        transforms.RandomResizedCrop(
            32, scale=(1.0 - 0.9 * strength, 1.0)),
        # No need to decay horizontal flip
        transforms.RandomHorizontalFlip(p=0.5),
        rnd_color_jitter,
        rnd_gray,
        transforms.ToTensor(),
    ])
    return transform
```

A.2 PSEUDO CODE OF DYNACL AND DYNACL++ ALGORITHM

Algorithm 1: DynACL algorithm

```
input  : Encoder $g$ and linear classification head $h$. Function of the augmentation set $\mathcal{T}(\cdot)$,
         number of pretraining epoch $T$, and post-processing epoch $T', T''$.
output : Pretrained and post-processed encoder $g$.
/* Phase 1: Momentum contrastive pretraining with
   augmentation annealing (DynACL)                                  */
forall $t \in \{1, 2, \cdots, T\}$ do
    for sampled mini-batch $\mathcal{X}$ do
        augment samples from $\mathcal{X}$ by augmentation set $\mathcal{T}(s(t))$
        $\mathcal{L} \leftarrow \mathcal{L}_{DynACL}(g; t)$
        update parameters in $g$ to minimize $\mathcal{L}$

/* Phase 2: Pseudo label based post-processing (DynACL++)           */
Extract the normal route of momentum encoder, denote it by $g_c$
$\{\hat{y}_i\} = \text{k\_means}(\{x_i\}; g_c)$
/* Post-processing phase 1, tune the head                           */
Extract and freeze the adversarial route of momentum encoder, denote it by $g_a$
forall epoch $\in \{1, 2, \cdots, T'\}$ do
    Standard-Training ($h \circ g_a, \{x_i\}, \{\hat{y}_i\}$)
/* Post-processing phase 2, tune the whole model                    */
Unlock the parameters in $g_a$
forall epoch $\in \{1, 2, \cdots, T''\}$ do
    TRADES ($h \circ g_a, \{x_i\}, \{\hat{y}_i\}$)
```

Algorithm 1 demonstrates the pseudo-code of our proposed DynACL and DynACL++. Note that DynACL++ adds a fast and simple post-processing phase to DynACL and enjoys higher robustness.

A.3 PIPELINE OF DYNACL AND DYNACL++

Figure 6 demonstrates the pipeline of the proposed DynACL and DynACL++. For DynACL (upper part of the figure), we adopt our proposed augmentation annealing strategy and the momentum

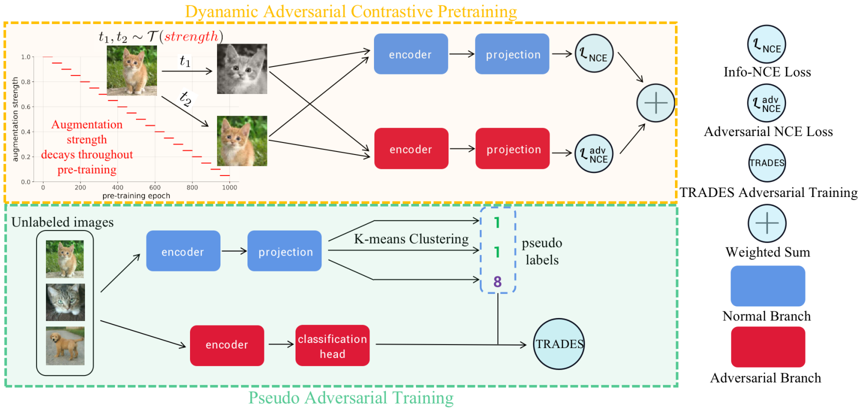


Figure 6: Pipeline of DynACL.

technique (He et al., 2020) in the pretraining phase. Projection heads are added on the top of both branches of the ResNet backbone. For simplicity, this architecture is not explicitly shown in the figure. For DynACL++, we add a simple and fast post-processing phase to DynACL (bottom part of the figure). Specifically, we replace the projection head with a linear classification head and adopt our PAT algorithm. Note that both two phases require **NO** true labels.

B DETAILS OF THE MEASUREMENT OF AUGMENTATION EFFECTS

B.1 CALCULATION PROTOCOLS

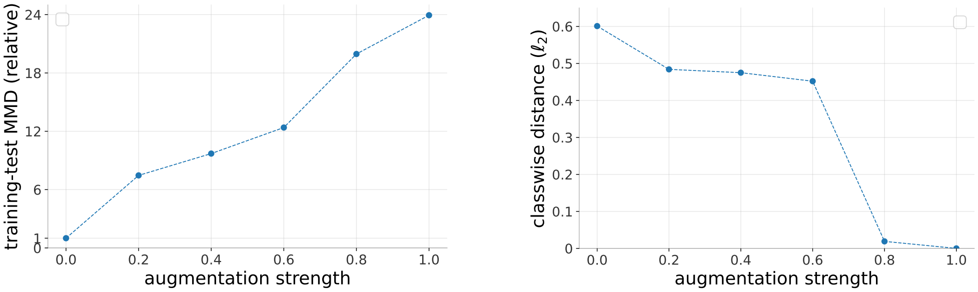
Calculation of minimal classwise distance. We randomly augment each sample from the training set 50 times and calculate the distance in terms of ℓ_∞ norm in the input space. We believe that the input space class separability is a vital indicator of adversarial robustness. Typical adversarial perturbation budget ε is defined in the input space, e.g., 8/255 under ℓ_∞-norm. Thus, given an augmented dataset D, whether the minimal input space classwise distance is smaller than $2 \cdot 8/255 = 16/255$ decides whether this dataset is adversarially separable, i.e., whether there exists a classifier that achieves 100% training robust accuracy.

Calculation of Maximum Mean Discrepancy (MMD). Denote the whole training set by $\mathcal{X}_{tr}$ and the whole test set by $\mathcal{X}_{ts}$. We first transform each set into a batch of tensors and then flatten them into two dimensions. Finally, we adopt Radical Basis Function as kernel function with bandwidth $10, 15, 20, 50$ to calculate the MMD between $\mathcal{X}_{tr}$ and $\mathcal{X}_{ts}$.

B.2 LATENT SPACE DISTRIBUTION GAP AND MINIMAL CLASSWISE DISTANCE

In Section 3.1, we have demonstrated the input space distribution gap between the training set and test set and found that stronger data augmentation will result in a wider distribution gap. To further justify our findings, we calculate the statistics using the distance of sample features in the latent space. Specifically, we utilize the Perceptual Similarity proposed in (Zhang et al., 2018) to calculate latent space distance.

Latent Space Distribution Gap. Specifically, for MMD, given perceptual similarity function $f(x, y)$ which takes two images x, y as inputs, the kernel function in MMD is defined as $1 - f(x, y)$. Note



(a) latent space distribution gap (b) latent space minimal classwise distance

Figure 7: (a) Latent space relative MMD between the augmented training set and clean test set. (b) Latent space minimal classwise distance on the training set.

that the perceptual similarity is lower for similar images and 0 for identical images. Figure 7(a) demonstrates the latent space MMD between the augmented training set and clean test set. Similar to input space MMD, stronger data augmentation will bring a bigger distribution gap, thus harming robustness generalization.

Latent Space Minimal Classwise Distance. Similarly, we can calculate the classwise distance using latent space distance. Specifically, we augment each sample from the training set 20 times and use a pretrained AlexNet to extract its latent space features. After normalization, We calculate the ℓ_2 distance between augmented samples from different classes and record the minimum. The results are shown in Figure 7(b). We can see that the latent classwise distance will also become significantly smaller under stronger augmentations (larger s), which generally agrees with the trend in the input space.

C EXPERIMENTAL DETAILS

C.1 PRETRAINING SETTINGS

Except that we adopt Simsiam Chen & He (2020) as the backbone, we follow ACL on other pretraining configurations. Specifically, we train the backbone for 1000 epochs with LARS optimizer and cosine learning rate scheduler. The projection MLP has layers. For DynACL pretraining, we set the decay period $K = 50$, and reweighting rate $\lambda = 2/3$. For DynACL++, we generate pseudo labels for training data by K-means clustering. First we tune the classification head only by stardard training for 10 epochs, then we tune both the encoder and the classification head by TRADES (Zhang et al., 2019) for 25 epochs.

C.2 EVALUATION PROTOCOLS

As for SLF and ALF we train the linear classifier for 25 epochs with initial learning rate 0.01 on CIFAR-10 and 0.1 on CIFAR-100 and STL-10 (decays at the 10th and 20th epoch by 0.1) and batch size 512. For AFF, we employ the TRADES (Zhang et al., 2019) loss with default parameters, except that in DynACL and DynACL++, we feed the clean samples to the normal encoder when generating adversarial perturbations. Finetuning lasts 25 epochs with initial learning rate 0.1 (decays at the 15th and 20th epoch by 0.1) and batch size 128. For all baselines and DynACL, we report the last result in the evaluation phase.

In the original implementation of AdvCL (Fan et al., 2021), they incorporate extra data from ImageNet to train a feature extractor based on SimCLR. For a fair comparison, we replace AdvCL's ImageNet-pretrained model with a CIFAR-pretrained model. This model is trained following the default hyperparameters and official implementations and reaches a clean accuracy of 88.15% on CIFAR-10 and 65.78% on CIFAR-100.

Table 6: Comparison between the threshold strategy and the original strategy.

	AA(%)	SA(%)
original ($1 \rightarrow 0$)	46.46	79.81
threshold at 0.3	45.89	80.14

D ADDITIONAL EXPERIMENTS

D.1 TRAINING DYNAMICS

Figure 8 demonstrates the training dynamics of ACL, DynACL, and supervised-AT (all with cosine learning rate schedule). We observe that the robust accuracy (PGD-20) of both DynACL and ACL increases continuously throughout the training process, while that of sup-AT suffers a significant decrease in the later period of training.

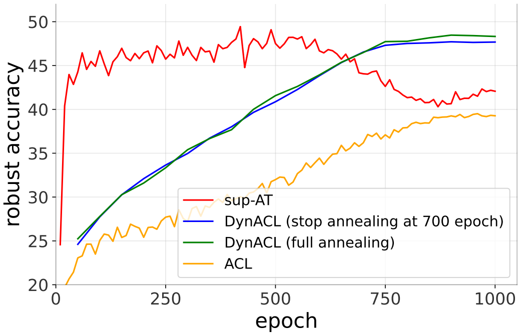


Figure 8: Training dynamics of ACL, DynACL, and supervised AT.

Figure 8 also depicts the effectiveness of annealing the augmentation strength s to 0. To verify this point, we consider an alternative schedule, where we threshold the augmentation at $s = 0.3$. In other words, we adopt a constant augmentation strength $s = 0.3$ after the 700th epoch. Table 6 shows that the thresholded strategy performs worse than the original strategy.

D.2 DATA AUGMENTATION STRENGTH IN SUP-AT

Popular sup-AT methods (Pang et al., 2020) typically adopt very weak data augmentations, including only horizontal flip and padding-crop. As shown in Figure 9, aggressive data augmentations are detrimental to the model's performance. As shown in Figure 4(c), this phenomenon also holds in self-AT.

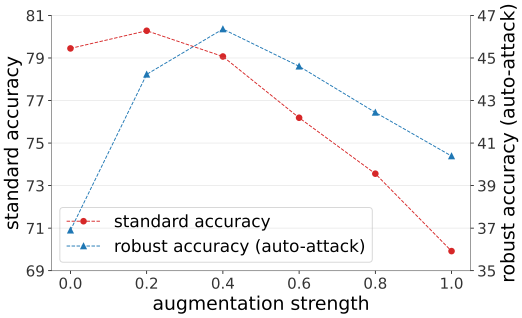


Figure 9: Performance of sup-AT with respect to augmentation strength s

Table 7: Ablation study augmentation annealing (Section 4.1) and momentum technique. Baseline method is ACL (Jiang et al., 2020), and the last line represent our proposed DynACL.

Annealing	Momentum	AA(%)	RA(%)	SA(%)
×	×	37.62	40.44	79.32
×	✓	36.43 **(-1.19)**	39.39 **(-1.05)**	74.80 **(-4.52)**
✓	×	44.92	48.65	75.76
✓	✓	45.04 **(+0.12)**	48.40 **(-0.25)**	77.41 **(+1.65)**

D.3 ABLATION OF AUGMENTATION ANNEALING AND MOMENTUM TECHNIQUE.

We further conduct an ablation study of augmentation annealing 4.1) and momentum technique in Table 7. Different from ACL which adopts SimCLR Chen et al. (2020a) as the backbone, our DynACL adopts SimSiam Chen & He (2020) as the backbone instead. We can see that momentum technique can bring a little bit benefit to our method.

E THEORETICAL JUSTIFICATION

In this section, following the theoretical analysis of adversarial training (Sinha et al., 2017), we provide a solid theoretical justification on our analysis. Specifically, we show that a large training-test distribution gap caused by aggressive data augmentations will indeed induce larger test errors.

Setup. Consider an input data space $\mathcal{X}$ and a bounded loss function $l(x;\theta)$ satisfying $|l(x;\theta)| \leq M$. For the ease of theoretical exposure, we consider the Wasserstein distance for measuring the distribution gap. In particular, the Wasserstein distance between P and Q is

$$W_c(P,Q) := \inf_{M\in\Pi(P,Q)} \mathbb{E}_M\left[c\left(X,X'\right)\right] \tag{10}$$

where $\Pi(P,Q)$ denotes the couplings of P and Q, and $c: X \to X$ defines a non-negative, lower semi-continuous cost function satisfying $c(x,x)=0, \forall\, x\in\mathcal{X}$. Accordingly, we adopt a robust surrogate objective $\phi_\gamma\left(\theta;x_0\right) := \sup_{x\in\mathcal{X}}\left\{\ell(\theta;x)-\gamma c\left(x,x_0\right)\right\}$, which allows adversarial perturbations of the data x in the Wasserstein space, modulated by the penalty γ. Considering the equivalence of norms in the finite-dimensional space, our discussion here in the Wasserstein space also sheds light on ℓ_p-norm bounded adversarial training.

Suppose the original data distribution (without augmentation) is P_{test}, and the augmented data distribution is P_{aug}. Denote the empirical augmented training data distribution as $\hat{P}_{\text{aug}}$ with n samples. Below, we are ready to establish theoretical guarantees on the out-of-distribution (OOD) generalization of *robust training* on the augmented data, *i.e.,* $\hat{P}_{\text{aug}}$ to the original test data distribution $\hat{P}_{\text{test}}$.

Theorem 1 *For any and a fixed $t>0$, the following inequality holds with probability at least $1-e^{-t}$, simultaneously for all $\theta\in\Theta, \rho\geq 0, \gamma\geq 0$:*

$$\sup_{P:W_c(P_{\text{test}},P_{\text{aug}})\leq\rho} \mathbb{E}_{P_{\text{test}}}[\ell(\theta;X)] \leq \gamma\rho + \mathbb{E}_{\widehat{P}_{\text{aug}}}\left[\phi_\gamma(\theta;X)\right] + \epsilon_n(t). \tag{11}$$

Here, $\epsilon_n(t) := \gamma b_1\sqrt{\frac{M}{n}}\int_0^1\sqrt{\log N\left(\mathcal{F}, M\epsilon, \|\cdot\|_{L^\infty(\mathcal{X})}\right)}d\epsilon + b_2 M\sqrt{\frac{t}{n}}$, where $\mathcal{F}$ is the hypothesis class, $N(\mathcal{F},\varepsilon,\|\cdot\|)$ is the corresponding cover number, and b_1 and b_2 are numerical constants.

Analysis. From Theorem 1, we can see that ρ controls the distribution gap between the original data distribution and the augmented data distribution, and a larger ρ indeed introduces a larger error term in the upper bound of the test performance. Specifically, when ρ is very large, *e.g.,* with aggressive augmentations, the test performance guarantees to be poor. This shows that theoretically, a too large training-test distribution gap introduced by aggressive augmentation will indeed contribute to worse generalization, which echos with our empirical inverstigation in Section 3.

Consequently, if we gradually anneal the augmentation strength s from 1 to 0, as we have done in the proposed DynACL, the distribution discrepancy ρ will gradually shrink (as it is nearly 0 when $s=0$

at last), leading to a smaller upper bound. Thus, our augmentation annealing strategy will indeed help bridge the distribution gap and improve generalization to test data.

E.1 PROOF OF THEOREM 1

We first state the following lemma from Sinha *et al.* (Sinha et al., 2017) that gives a duality result between the objectives on any two distributions P and Q.

Lemma 2 *Let* $\ell : \Theta \times \mathcal{X} \to \mathbb{R}$ *and* $c : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$*be continuous. Let* $\phi_\gamma(\theta; z_0) = \sup_{z \in \mathcal{X}} \{\ell(\theta; z) - \gamma c(z, z_0)\}$ *be the robust surrogate* (2 b)*. For any distribution* Q *and any* $\rho > 0$*,*

$$\sup_{P: W_c(P,Q) \le \rho} \mathbb{E}_P[\ell(\theta; X)] = \inf_{\gamma \ge 0} \{\gamma\rho + \mathbb{E}_Q[\phi_\gamma(\theta; X)]\}$$

and for any $\gamma \ge 0$*, we have*

$$\sup_P \{\mathbb{E}_P[\ell(\theta; X)] - \gamma W_c(P, Q)\} = \mathbb{E}_Q[\phi_\gamma(\theta; X)].$$

Applying Lemma 2 to our problem, we have the following deterministic result

$$\sup_{P_{\text{test}}: W_c(P_{\text{test}}, P_{\text{aug}}) \le \rho} \mathbb{E}_{P_{\text{test}}}[\ell(\theta; X)] \le \gamma\rho + \mathbb{E}_{P_{\text{aug}}}[\phi_\gamma(\theta; X)]$$

for all $\rho > 0$, distributions P_{aug}, and $\gamma \ge 0$. Next, we show that $\mathbb{E}_{\widehat{P}_{\text{aug}}}[\phi_\gamma(\theta; X)]$ concentrates around its population counterpart at the usual rate. Also remind that we have that $\phi_\gamma(\theta; z) \in [-M, M]$, because $-M \le \ell(\theta; x) \le \phi_\gamma(\theta; x) \le \sup_x \ell(\theta; x) \le M$. Thus, the functional $\theta \mapsto F_n(\theta)$ satisfies bounded differences, and applying standard results on Rademacher complexity and entropy integrals gives the result.