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 methods 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.

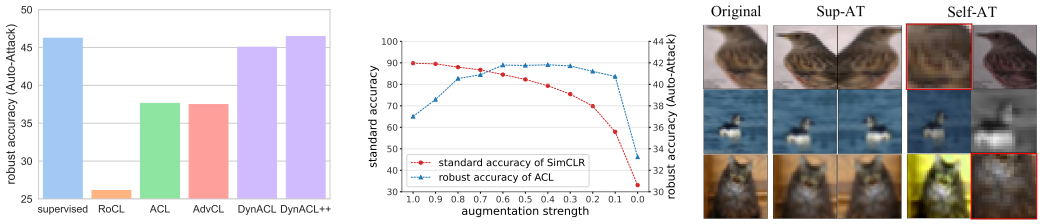
1 INTRODUCTION

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 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 which 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 (HaoChen et al., 2021; Wang et al., 2022), 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 standard contrastive learning (SimCLR) 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 the



(a) robustness comparison (b) 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) and adversarial contrastive learning (Jiang et al., 2020) 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.

adversarial robustness of ACL. This reveals a critical dilemma of self-AT, *i.e.,* the default 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 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 DynACL++. As a preview of the 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 for 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):

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

Here $\text{sim}(\cdot, \cdot)$ 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., 2020; 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,

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

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:

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

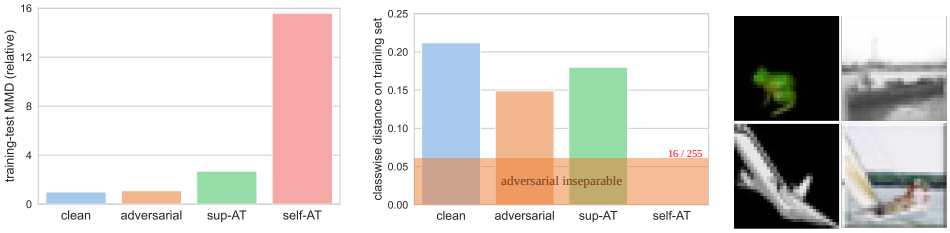
After training, they use the adversarial branch g_a as the robust encoder. They empirically show that ACL obtains better robustness than the single branch version. Built upon ACL, AdvCL (Fan et al., 2021) further leverages an additional ImageNet-pretrained model to generate pseudo-labels for sup-AT via 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). For intuitive exposure, we mainly resort to empirical evidence on real-world data. Rigorously, we also provide theoretical justifications for these phenomena in Appendix E.

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



(a) training-test distribution gap (b) classwise distance (c) worst-case samples

Figure 2: (a & b) Comparison of different augmentation strategies on CIFAR-10, where "clean" denotes no augmentation; "adversarial" denotes adversarial examples generated by PGD attack; (c) Training examples from different classes of CIFAR-10 that could yield pure black or pure white augmented views.

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 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 the 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 The Dilemma of Self-AT under Static Augmentation Strength

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

Quantifying Augmentation Strength. In order to give a quantitative measure of the augmentation strength, 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. See a detailed configuration of $\mathcal{T}(s)$ in Appendix A.1.

Static Augmentation Cannot Fully Resolve the Dilemma of Self-AT. The discussion in Section 3.1 reveals that strong augmentations are harmful to adversarial training while being critical for representation learning (Figure 1(b), Appendix D.3 for more details). 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

[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.

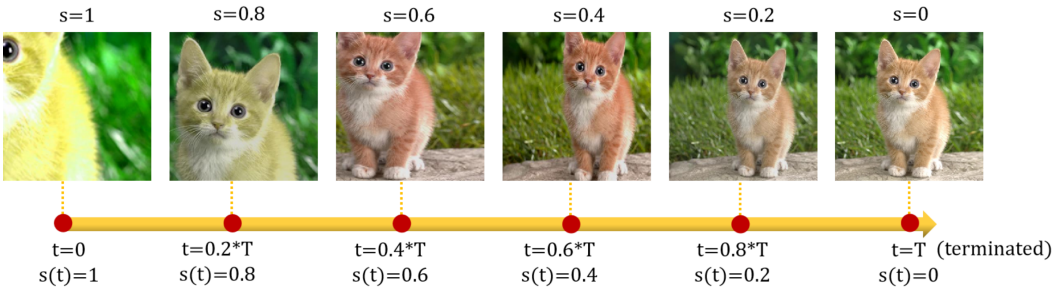
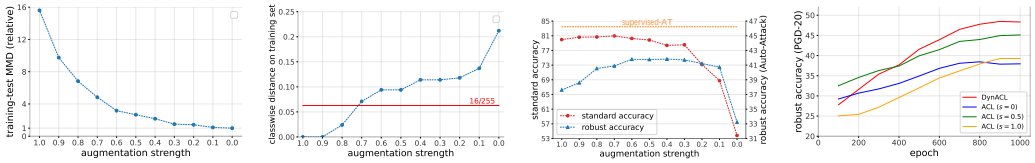


Figure 3: Illustration of augmenting the same image of a cat with different augmentation strengths $s \in [0, 1]$, where a smaller s (left to right) indicates milder image distortion.



(a) training-test distribution gap (b) 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.

and eliminates class inseparability, which benefits adversarial robustness (blue line in Figure 4(c)) and hurts standard accuracy at the same time (the 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 the 46.23% robustness of sup-AT (orange line in Figure 4(c)). This suggests that the dilemma of augmentation strength cannot be fully resolved by tuning a static augmentation strength. The training process in Figure 4(d) presents that strong or weak augmentations have their own defects while medium augmentations only bring a limited gain (green line). In the next section, we will introduce a new dynamic augmentation strategy to resolve this dilemma (a preview of performance is shown as the 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 of self-AT augmentations. In Section 4.1, we introduce our dynamic pretraining method for self-AT, named DynACL. In Section 4.2, we further propose an improved version of DynACL with a fast post-processing stage to mitigate the gap between 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

In view of the dilemma of static augmentation strength (Section 3.2), we propose a new *dynamic* training recipe for self-AT, named Dynamic Adversarial Contrastive Learning (**DynACL**), which consists of a dynamic schedule for data augmentation and a dynamic objective for model training.

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, we can enforce self-AT to learn meaningful representations by adopting aggressive augmentations with a large augmentation strength $s(t)$. Afterwards, we could gradually mitigate the training-test distribution gap and class inseparability by annealing down the augmentation strength $s(t)$ (see Figures 4(a) and 4(b)). With this dynamic augmentation schedule, we could combine the best of both worlds: on the one hand, we can benefit from the representation ability of contrastive learning at the beginning stage; on the other hand, we can achieve better adversarial robustness by closing the distribution gap at the later stage.

Following the analysis above, we design 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, this schedule $s(t)$ starts from strong augmentations with $s = 1$ at the start epoch, and gradually decays to a weak augmentation with $s = 0$ by reducing K/T every K epochs.

Dynamic ACL Loss. Following the same idea above, we can also design a dynamic version of the ACL loss in Eq. 4. Instead of adopting an equal average of the clean and adversarial contrastive losses, we can learn meaningful representations in the beginning stage with the clean contrastive loss ℓ_{NCE} (Eq. 2). Later, we can gradually turn our focus to adversarial robustness by down-weighting the clean contrastive loss and up-weighting the adversarial loss ℓ_{AdvNCE} (Eq. 3). Following this principle, we propose a dynamic ACL loss with a dynamic reweighting function $w(t)$ for the t-th epoch:

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

For simplicity, we assign the dynamic reweighting function $w(t)$ as a function of the dynamic augmentation schedule $s(t)$:

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

where $\lambda \in [0, 1]$ denotes the reweighting rate. In this way, when the training starts ($t = 0$), we adopt an equal combination of the clean and adversarial losses as in ACL. As training continues, the augmentation strength $s(t)$ anneals down, and the reweighting coefficient $w(t)$ becomes larger. Accordingly, the clean loss ℓ_{NCE} will have a smaller weight while the adversarial loss ℓ_{AdvNCE} will have a larger weight. At last ($t = T$), the weight of clean loss is decreased to $1 - \lambda$, and the weight of adversarial loss is increased to $1 + \lambda$. A larger reweighting rate λ indicates more aggressive reweighting. Our dynamic ACL loss degenerates to vanilla ACL loss with $\lambda = 0$ as a special case.

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 lying 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.

To achieve this, we propose a simple post-processing phase with Pseudo Adversarial Training (PAT) to mitigate this "task gap". Given a pretrained encoder with a clean branch g_c and an adversarial branch g_a, we simply generate pseudo labels $\{\hat{y}_i\}$ by k-means clustering using the clean branch g_c, which natural accuracy is relatively high because it is fed with natural samples during training. Inspired by the Linear Probing then full FineTuning (LP-FT) strategy (Kumar et al., 2022), we propose to learn a classifier with Linear Probing then *Adversarially* full FineTuning (LP-AFT) for better feature robustness: we first use standard training to train a linear head $h : \mathbb{R}^m \to \mathbb{R}^k$ (k is the number of pseudo-classes) with fixed adversarial branch g_a, and then adversarially finetune the entire classifier $h \circ g_a$ on generated pseudo pairs $(x, \hat{y})$ using the TRADES loss (Zhang et al., 2019). In practice, we only perform PAT for a few epochs (*e.g.,* 25), 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 and name the improved version DynACL++.

Dicsusions. Our DynACL is built upon the dual-stream framework of ACL (Jiang et al., 2020). Compared to ACL with static augmentation strength and learning objective, DynACL adopts a dynamic augmentation schedule together with a dynamic learning objective. Also built upon ACL, AdvCL (Fan et al., 2021) leverages a third view for high-frequency components and pseudo labels into the entire pretraining phase, which brings significant computation overhead. In comparison, we devise the LP-AFT process to align linear encoders with pretrained features and exploit self-generated pseudo labels to mitigate the task gap. Compared to AdvCL, our method brings more significant improvements whil being much more computionally efficient.

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-based 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). Additional 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). We set the decay period $K = 50$, and reweighting rate $\lambda = 2/3$. As for other parts, we mainly follow the training recipe of ACL (Jiang et al., 2020), while we additionally borrow two training techniques from modern contrastive learning variants: the stop gradient operation and the momentum encoder. Both are proposed by BYOL (Grill et al., 2020) and later incorporated in MoCo-v3 (Chen et al., 2021) and many other methods. An ablation study on these two components can be found in Appendix D.2, where we show that the improvement of DynACL mainly comes from the proposed dynamic training strategy.

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. We evaluate the robustness of sup-AT and self-AT methods on CIFAR-10, CIFAR-100, and STL-10 in Table 1. As we can see, DynACL outperforms all previous self-AT methods in terms of robustness without using additional data[2]. Specifically, DynACL brings a giant 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%) under the auto-attack benchmark. Meanwhile, we notice that DynACL alone leads to a slight decrease in standard accuracy in CIFAR-10 and STL-10, suggesting that annealed augmentations at a later stage might slightly hurt the discriminative power to trade off robustness. Nevertheless, post-processing in DynACL++ can largely mitigate this issue and improve standard accuracy and robustness. As a result, DynACL++ achieves even higher AA accuracy than the supervised vanilla AT (with heavily tuned default hyperparameters (Pang et al., 2020)) on CIFAR-10. It is the first time 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 the 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.

[2]Note that AdvCL incorporates extra data as 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)".

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)	42.57	80.85	42.54	79.41	49.97	83.27

Table 3: Semi-supervised AT on CIFAR-10. RA stands for robust accuracy under the PGD-20 attack.

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**

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 method ACL Jiang et al. (2020). The results are shown in Table 3. We observe that our DynACL++ can demonstrate fairly good performance even when only 1% of labels are available.

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 the 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 100th 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.

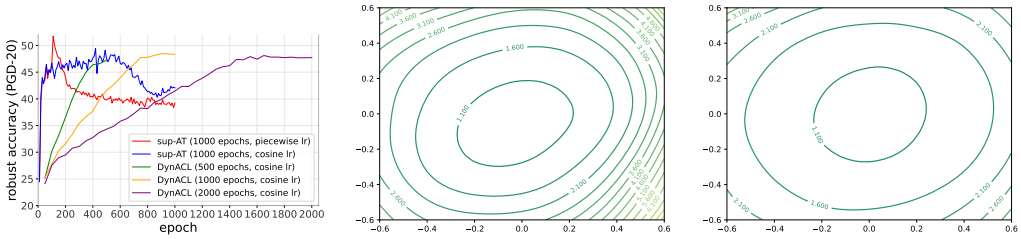
Loss Landscape. Previous studies (Prabhu et al., 2019; Yu et al., 2018; Wu et al., 2020) have shown that a flatter weight landscape often leads to better robust generalization. Motivated by that discovery, we visualize the loss landscape of ACL and DynACL++ following (Li et al., 2018). Figure 5(b) and 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 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, DynACL's dynamic strategy (augmentation and loss) introduces little computation overhead, and it spends even less time due to the stop gradient on the target branch. Moreover, since the post-processing only lasts a few epochs, DynACL++ is also faster

[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 4: Ablation study two key designs of DynACL++ on CIFAR-10. 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 **(-1.92)**
✓	×	45.04 **(+7.42)**	48.40 **(+7.96)**	77.41 **(-1.91)**
✓	✓	46.46 **(+8.84)**	49.21 **(+8.77)**	79.81 **(+0.49)**



(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.

than ACL. In comparison, AdvCL is much slower and even triples the training time of 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 significantly outperforms the constant one. Also, we notice that the step-wise annealing strategy performs slightly better. This may be 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 $\lambda = 2/3$.

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**

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 simple trade-off by selecting a medium augmentation strength, we propose to adopt a dynamic 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*, 2020b.

Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *CVPR*, 2021.

Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *ICCV*, 2021.

Tsz-Him Cheung and Dit-Yan Yeung. Adaaug: Learning class-and instance-adaptive data augmentation policies. In *ICLR*, 2021a.

Tsz-Him Cheung and Dit-Yan Yeung. Adaaug: Learning class-and instance-adaptive data augmentation policies. In *ICLR*, 2021b.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *AISTATS*, 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. *JMLR*, 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*, 2014.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *JMLR*, 2012.

Jean-Bastien Grill, Florian Strub, Florent Altché, C. Tallec, Pierre H. Richemond, Elena Buchatskaya, C. Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: A new approach to self-supervised learning. In *NeurIPS*, 2020.

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.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv*, 2022.

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*, 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*, 2018.

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

Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and John Whaley. Understanding adversarial robustness through loss landscape geometries. *arXiv*, 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*, 2017.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *ECCV*, 2020.

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. In *ICLR*, 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*, 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.

Fengwei Zhou, Jiawei Li, Chuanlong Xie, Fei Chen, Lanqing Hong, Rui Sun, and Zhenguo Li. Metaaugment: Sample-aware data augmentation policy learning. In *AAAI*, 2021.

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:

```
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 the momentum encoder, and 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 the 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 technique (He et al., 2020) in the pretraining phase. Projection heads are added on the top of both

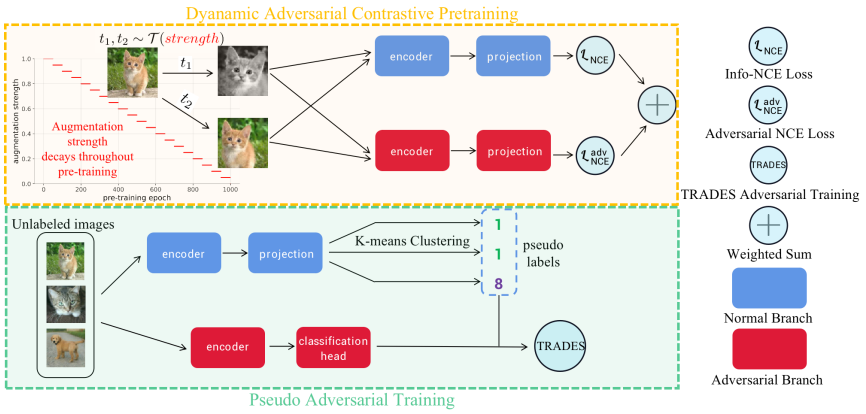


Figure 6: Pipeline of DynACL.

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

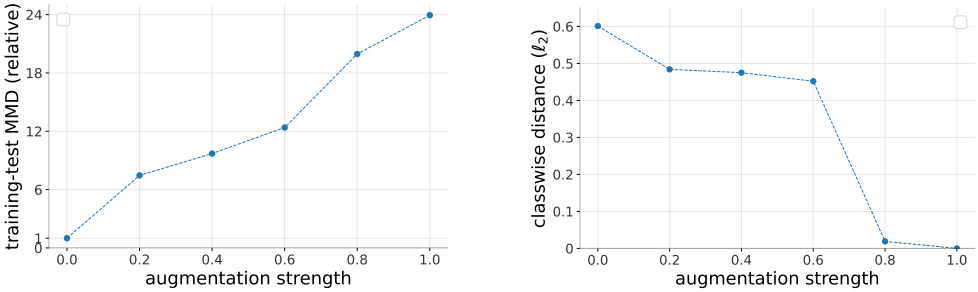
Calculation of minimal classwise distance. We randomly augment each sample from the training set 50 times and calculate the distance in terms of ℓ_∞ norm in the input space. We believe that the input space class separability is a vital indicator of adversarial robustness. Typical adversarial perturbation 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 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



(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.

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 the momentum backbone and stop-gradient operation (Grill et al., 2020), 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 3 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 standard 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 an initial learning rate of 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 an initial learning rate of 0.1 (decays by 0.1 at the 15th and 20th epoch) 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.

D ADDITIONAL EXPERIMENTS

D.1 VISUALIZATION OF 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.

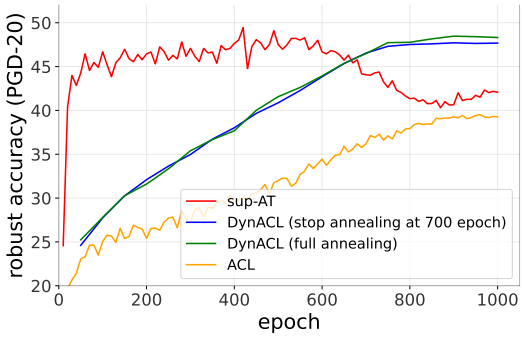


Figure 8: Training dynamics of ACL, DynACL, and supervised AT.

Figure 8 also depicts the effectiveness of annealing the augmentation strength s to 0. To further examine this point, we consider an alternative schedule, where we threshold the augmentation strength when it reaches $s = 0.3$ at the 700-th epoch, and adopt a constant augmentation strength $s = 0.3$ afterwards. Table 6 shows that the thresholding strategy performs a little bit worse than the original strategy.

Table 6: Comparison between different augmentation strategies.

Augmentation strategy	AA(%)	SA(%)
Fixed at 1 (ACL)	37.62	79.32
Anneal from 1 to 0	**46.46**	79.81
Anneal from 1 to 0.3	45.89	**80.14**

D.2 ADDITIONAL ABLATION STUDY

Here we conduct an ablation study on the momentum encoder and stop-gradient operation that is additional adopted by DynACL. We can see without both techniques, DynACL still achieves very high AA robustness (44.9%) that is significantly better than ACL (Jiang et al., 2020). As for the two techniques, we notice that the stop gradient alone has limited improvement, also it helps improve training efficiency. The momentum encoder, instead, help improves the standard accuracy by nearly 2%, while attains similar AA robustness to the vanilla DynACL (ACL + dynamic strategy).

Table 7: Ablation study of momentum backbone and stop gradient operation on the CIFAR-10 dataset. Our DynACL (last line) uses both momentum encoder and stop gradient operation following (Grill et al., 2020). Baseline method (first line) is ACL (Jiang et al., 2020).

Dynamic Strategy	Stop Gradient	Momentum	AA(%)	RA(%)	SA(%)
×	×	×	37.62	40.44	**79.32**
✓	×	×	44.92	**48.65**	75.76
✓	✓	×	45.02	48.57	75.00
✓	✓	✓	**45.04**	48.40	77.41

D.3 Further Investigations on Data Augmentations

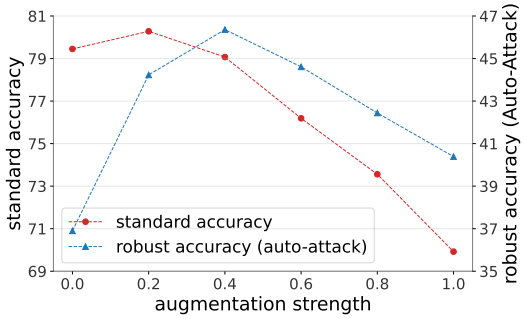


Figure 9: Performance of sup-AT with respect to augmentation strength s on CIFAR-10.

Effect of Augmentation Strength in Sup-AT. Popular sup-AT methods (Zhang et al., 2019; 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.

Table 8: Comparative results of AdaAug (Cheung & Yeung, 2021a) on CIFAR-10.

Method	AA(%)	SA(%)
ACL	37.62	79.32
AdaAug	37.91	**80.57**
DynACL++	**46.46**	79.81

Comparison among Augmentation Strategies. Data Augmentation is an effective approach to enhance data diversity and prevent overfitting. Canonical augmentations are input-agnostic and static across the training process. Recently, a few works have explored class-wise or instance-wise adaptive augmentation strategies, such as AdaAug (Cheung & Yeung, 2021b) MetaAugment (Zhou et al., 2021). We instead explore an augmentation schedule that dynamically changes during the *training process*. Compared to meta-learning strategies that bring additional training costs, our augmentation schedule is simple and does not introduce training or parameter overhead.

In practice, we also compare our method against AdaAug (Cheung & Yeung, 2021a). We observe that it can bring small improvements, but is much less effective to DynACL++ (Jiang et al., 2020) (Table 8). This shows that our dynamic strategy is very suitable for the self-AT task compared to general-purpose adaptive augmentations.

Table 9: Effect of data augmentations in self-ST (SimCLR) on CIFAR-10.

	Constant						Dynamic
Augmentation strength s	0	0.2	0.4	0.6	0.8	1.0	$1 \rightarrow 0$
SA(%)	33.07	69.86	79.34	84.53	87.99	89.82	85.92

Effect of Augmentation Strength in Self-supervised Standard Training. In Table 9, we provide the effect of data augmentation strength in sup-ST. It shows that weaker data augmentations will bring worse performance, and our proposed dynamic schedule may also yield performance degradation. This phenomenon demonstrates the difference between self-ST and self-AT, which is consistent with our discussion in Section 3.2.

D.4 Generalization Ability to Other Methods

Applicability on Other Self-AT Methods. In Table 4, we evaluate the effectiveness of our strategy when built upon ACL. Here, we further apply it to two other self-AT backbones, RoCL (Kim et al.,

Table 10: Comparison results of applying our training strategy to other self-AT frameworks on CIFAR-10.

Method	AA(%)	SA(%)
RoCL	26.12	**77.90**
+ DynACL++	**39.72**	74.84
AdvCL	37.46	73.23
+ DynACL++	**41.27**	**74.00**

2020) and AdvCL (Fan et al., 2021). Specifically, we apply the dynamic augmentation schedule and the post-processing stage in our implementation. Results are shown in Table 10. We can see that our DynACL++ also attains significant improvements in robustness when applied to RoCL and AdvCL.

Table 11: Comparison results on CIFAR-10 with different backbone architectures, and different SSL frameworks.

(a) Self-AT with AlexNet backbone

Method	AA(%)	SA(%)
ACL	11.79	54.87
DynACL++	**23.93**	56.61

(b) Self-AT with SimSiam framework

Method	AA(%)	SA(%)
ANL	32.02	67.05
DynANL++	**36.09**	66.61

Applicability with Different Backbone Networks. We also compare ACL and our DynACL++ on a different network, AlexNet. From Table 11(a), we can see that our method brings 12% absolute improvements in AA robustness, which is over 100% relative improvement over ACL.

Applicability with Other SSL Frameworks. We also show that our method can be applied to other SSL frameworks, such as non-contrastive learning, for example, BYOL (Grill et al., 2020), SimSiam (Chen & He, 2021). Here we take SimSiam as an example. Its learning objective is solely a cosine similarity loss between positive pairs, and it utilizes a predictor and stop gradient to avoid feature collapse. Formally, the SimSiam loss is

$$\ell_{\text{Sim}}(x, x^+; f, g) = 1 - \text{sim}(f(g(x)), \text{sg}(g(x^+))), \tag{8}$$

where f denotes a predictor module (*e.g.,* an MLP), and sg denotes the stop gradient operator. We adapt it to the existing ACL framework by simply replacing the (both attack and learning) loss function of ACL from the InfoNCE loss to the SimSiam similarity loss above. Specifically, the adversrial similarity loss is adopted for generating adversarial examples:

$$\ell_{\text{AdvSim}}(x, x^+; f, g) = \max_{\delta, \delta^+ \in \Delta} \ell_{\text{Sim}}(x + \delta, x^+ + \delta^+; f, g). \tag{9}$$

Afterwards, we train the model with the following Adversarial Non-contrastive Learning (ANL) loss, which simply replaces the contrastive loss with the similarity loss in the ACL loss (Eq. 4):

$$\ell_{\text{ANL}}(x, x^+; f, g) = \ell_{\text{Sim}}(x, x^+; f_c, g_c) + \ell_{\text{AdvSim}}(x, x^+; f_a, g_a), \tag{10}$$

where f_c, g_c denote the clean branch, and f_a, g_a denote the adversarial branch. All the other configurations are kept the same as ACL.

Dynamic Version of Adversarial Non-contrastive Learning. When applying our dynamic strategy to ANL (dubbed DynANL), we also modify it in the same way as to ACL by adopting a dynamic augmentation schedule (Eq. 5) and a dynamic ANL loss:

$$\mathcal{L}_{\text{DynANL}}(g; t) = (1 - w(t))\mathcal{L}_{\text{Sim}}(g_c; s(t)) + (1 + w(t))\mathcal{L}_{\text{AdvSim}}(g_a; s(t)), \tag{11}$$

and $w(t)$ is kept the same as Eq. 7. After pretraining, we can apply the same post-processing stage (Section 4.2) to DynANL, and get DynANL++.

Results. We compare ANL and DynANL++ on CIFAR-10 with ResNet-18 backbone. As shown in Table 11(b), DynANL++ still obtains a significant improvement (4% AA) over the vanilla ANL, demonstrating that our dynamic strategy is also effective for other SSL frameworks.

E THEORETICAL ANALYSIS ON THE EFFECT OF AUGMENTATION STRENGTH

In Section 3, we have analyzed the effect of augmentation strength on self-AT from an empirical perspective. In this section, we further provide theoretical justifications on how strong augmentations will bring training-test distribution gap (Section E.1) and smaller classwise distance (Section E.2).

E.1 EFFECT ON TRAINING-TEST DISTRIBUTION GAP

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

$$W_c(P,Q) := \inf_{M \in \Pi(P,Q)} \mathbb{E}_M \left[c\left(X, X'\right)\right] \tag{12}$$

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 $\theta \in \Theta, \rho \geq 0, \gamma \geq 0$:*

$$\sup_{P: W_c(P_{\text{test}}, P_{\text{aug}}) \leq \rho} \mathbb{E}_{P_{\text{test}}}[\ell(\theta; X)] \leq \gamma\rho + \mathbb{E}_{\widehat{P}_{\text{aug}}}\left[\phi_\gamma(\theta; X)\right] + \epsilon_n(t). \tag{13}$$

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 echoes with our empirical investigation 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.1.1 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 $\phi_\gamma\left(\theta; z_0\right) = \sup_{z \in \mathcal{X}} \left\{\ell(\theta; z) - \gamma c\left(z, z_0\right)\right\}$ be the robust surrogate* (2 b)*. For any distribution Q and any $\rho > 0$,*

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

and for any $\gamma \geq 0$, we have

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

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

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

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

E.2 Effect on Classwise Distance

Setup. Without loss of generality, we consider a binary classification task ($y = \{0, 1\}$) with *separable* data $\bar{D}_0 = \{\bar{x}_i^0\}$ from class $y = 0$ and $\bar{D}_1 = \{\bar{x}_j^1\}$ from class $y = 1$. Denote the classwise distance as the the minimal ℓ_∞ distance between all cross-class pairs,

$$d = \min_{\bar{x}_i^0 \in \bar{D}_0, \bar{x}_j^1 \sim \bar{D}_1} \|\bar{x}_i^0 - \bar{x}_j^1\|_\infty. \tag{14}$$

Because the data are classiwise spearable, we have $d > 0$. Then, we consider a simple data augmentation scheme, that adds a small noise δ uniformly drawn from a ℓ_∞-ball $\mathcal{B}_\varepsilon = \{\delta | \|\delta\| \leq \varepsilon\}$ with norm ε to each raw sample $\bar{x}$, resulting in an augmented sample $x = \bar{x} + \delta$. Here, the norm ε controls the augmentation strength: a large ε adds stronger perturbations to the original sample.

Theorem 3 *Denote the classwise distance of the augmentated data with strength ε as d_ε, we have*

$$d_\varepsilon \leq d - 2\varepsilon. \tag{15}$$

Therefore, as augmentations become stronger (larger ε), the classwise distance will shrink. When augmentations become too strong ($\varepsilon > d/2$), the classwise distance becomes zero, i.e., $d = 0$.

Theorem 3 shows that a strong augmentation strength will indeed cause smaller classwise distance, or even make the data inseparable, which echos our empirical observations in Section 3.1.

E.2.1 Proof of Theorem 3

Denote a sample pair that attains the smallest classwise distance as $(\bar{x}_*^0, \bar{x}_*^1)$, *i.e.,* $d = \|\bar{x}_*^0 - \bar{x}_*^1\|_\infty$. Then, after augmentation, the classwise distance d_ε satisfies

$$\begin{aligned} d_\varepsilon &= \min_{x_i^0 \in \bar{D}_0^\varepsilon, \bar{x}_j^1 \sim \bar{D}_1^\varepsilon} \|x_i^0 - x_j^1\|_\infty && (16) \\ &\leq \min_{\delta_0, \delta_1 \in \mathcal{B}_\varepsilon} \|(x_*^0 + \delta_0) - (x_*^1 + \delta_1)\|_\infty && (17) \\ &= d - 2\varepsilon, && (18) \end{aligned}$$

where $D_0^\varepsilon = \{\bar{x}_i^0 + \delta | \bar{x}_i^0 \in \bar{\mathcal{D}}_0, \delta \in \mathcal{B}_\varepsilon\}$ and $D_1^\varepsilon = \{\bar{x}_j^1 + \delta | \bar{x}_j^0 \in \bar{\mathcal{D}}_1, \delta \in \mathcal{B}_\varepsilon\}$ denote the classwise augmented data with augmentation strength ε.