Adversarial Momentum-Contrastive Pre-Training

Cong Xu[a,1], Dan Li[a,2], Min Yang[a,*]

[a]School of Mathematics and Information Sciences, Yantai University, Yantai 264005, China

Abstract

Recently proposed adversarial self-supervised learning methods usually require big batches and long training epochs to extract robust features, which will bring heavy computational overhead on platforms with limited resources. In order to help the network learn more powerful feature representations in smaller batches and fewer epochs, this paper proposes a novel adversarial momentum contrastive learning method, which introduces two memory banks corresponding to clean samples and adversarial samples, respectively. These memory banks can be dynamically incorporated into the training process to track invariant features among historical mini-batches. Compared with the previous adversarial pre-training model, our method achieves superior performance with smaller batch size and less training epochs. In addition, the model outperforms some state-of-the-art supervised defensive methods on multiple benchmark datasets after being fine-tuned on downstream classification tasks.

Keywords: adversarial robustness, contrastive learning, memory bank, fine-tuning

1. Introduction

Although deep neural networks (DNNs) have achieved state-of-the-art performance on many challenging computer vision tasks, it was soon found that they are extremely vulnerable to semantic invariant corruptions [9, 23, 29] or adversarial attacks [10, 30], which means that very small perturbations on the original input could cause the network to make wrong prediction.

To remedy this deficiency, robust feature learning has attracted the attention of researchers [7, 19]. Among them, supervised adversarial training [10, 21, 31, 35], which guides the learning of neural networks with additional adversarial samples, has become the most popular method. Recently, some unsupervised robust learning methods [4, 15] have also been developed. The main idea is to treat adversarial samples as a special kind of data augmentations, and then adopt some self-supervised learning framework to extract the robust features contained in the data. However, unlike ordinary data augmentations, which are usually determined before training, adversarial examples are dynamically generated through training iterations and are greatly influenced by the current network parameters. So in order to capture the invariant features among them, very large batch sizes and long training epochs must be used [4, 15], which will bring heavy computational overhead on platforms with limited resources.

In this work, we aim to address the above problem by developing a novel **A**dversarial **MO**mentum-**C**ontrastive (AMOC) learning approach, which can leverage the historical information of mini-batches to help extract robust feature representations in a more efficient manner. More specifically, we build two memory banks to track clean and adversarial feature representations that are consistent across different mini-batches, respectively. The clean memory bank provides negative keys from clean data for contrastive learning, thus guaranteeing the natural accuracy of the model, whereas the adversarial memory bank provides negative keys from historical adversarial examples, thereby avoiding the learning inconsistency problem caused by the dynamic changes of the adversarial samples.

It is worthy to point out that in the original contrastive framework [12] there is only one clean memory bank for standard training. However, in adversarial training, we need not only clean samples but also their adversarial counterparts, and the distributions of clean and adversarial samples are quite different, so it is necessary to build separate memory banks for clean and adversarial samples, respectively. With the aid of the sperate memory banks, the network can better extract the invariant features contained in the perturbed data. Our experimental results demonstrate that using separate memory banks does perform better than a single one. Moreover, extensive experiments show that the proposed method even outperforms some state-of-the-art supervised defensive methods on multiple benchmark datasets after being fine-tuned on downstream classification tasks.

The main contributions of this work are summarized as follows:

- We give insights into why current adversarial contrastive learning algorithms require large batch sizes and long training epochs to reach convergence. Based on this, we explain the necessity of introducing separate memory banks to learn a better query encoder.
- We develop a novel contrastive learning framework named AMOC, which integrates a standard clean memory bank with an additional adversarial memory bank. And we also

*Corresponding author: yang@ytu.edu.cn
[1]Email: congxueric@gmail.com
[2]Email: danliai@hotmail.com

show how to train the framework efficiently in an end to end manner.

- We analytically identify a good form of training loss for AMOC and empirically discuss the deficiencies of other loss forms, with particular emphasis on comparison with the single memory bank mode.
- Through extensive empirical evaluation, we show that AMOC is an effective defense strategy against a wide range of recently proposed state-of-the-art attacks in the literature. Compared with the existing contrastive pre-training methods, the proposed approach exhibits superior robustness and better computational efficiency. After fine-tuning on classification tasks, AMOC can successfully outperform many supervised adversarial defenses on the widely used datasets.

The remainder of the paper is organized as follows. Section 2 introduces the related works regarding adversarial robustness and contrastive learning. Section 3 elaborates the research motivation and introduces the framework of AMOC. Section 4 gives a comprehensive experimental results to demonstrate the efficiency of the proposed method. Finally, the conclusion is drawn in Section 5.

2. Related Works

2.1. Attack models

Since the discovery that neural networks are vulnerable to artificial perturbations, a series of attack methods have emerged to test the robustness of networks [2, 5, 24, 21, 30, 31]. These attack models craft small perturbations to clean samples to generate various adversarial examples. Fast gradient sign method (FGSM) [10] is a simple yet effective attack method that utilizes the sign of the gradients of the loss function to generate adversarial samples. Projected gradient descent (PGD) [21] is a more powerful iterative attack that starts from a random position in the neighborhood of a clean input and then applies FGSM for several iterations. DeepFool [24] is a gradient-based attack algorithm that iteratively linearizes the classifier to generate the smallest perturbation sufficient to change the classification label. C&W [2] is one of the most powerful attack to detect adversarial samples in ℓ_2 norm. Sparse ℓ_1 descent (SLIDE) [31] is a more efficient model that overcomes the inefficiency of PGD in searching for ℓ_1 perturbations. Recently, Croce et al. [5] combined four diverse attacks into a more aggressive one, AutoAttack, as a new benchmark for empirical robustness evaluation

2.2. Adversarial training

To enhance the robustness of neural networks against adversarial attacks, numerous defense methods have been developed from different perspectives [7, 14, 19, 26, 34]. Among them, adversarial training [10, 21, 35], which minimizes the worst-case loss in the perturbation region, is considered to be one of the most powerful defenses. For example, given a distribution $\mathcal{D}$ over samples x and labels y, standard adversarial training [21] optimizes the following objective:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}} \max_{\|\delta\|\leq\epsilon} \mathcal{L}(x+\delta, y; \theta),$$

while tradeoff-inspired adversarial defense (TRADES) [35] considers a trade-off between natural accuracy and adversarial robustness:

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}} \left(\mathcal{L}(x, y; \theta) + \frac{1}{\lambda} \max_{\|\delta\|\leq\epsilon} \mathcal{L}(x, x+\delta; \theta) \right).$$

Nonetheless, adversarial training-based methods suffer from heavy computational overhead and are not scalable. The generality and reusability of pre-trained models can alleviate this problem to some extent, since robust models required for various downstream tasks can be obtained after simple fine-tuning.

2.3. Contrastive learning

Contrastive learning [3, 12, 25] is a popular self-supervised learning framework, which maximizes the similarity of a sample to its distinct views and minimizes its similarity with other instances. Different contrastive learning approaches usually adopt different strategies to generate various views and negative keys. One of the simple yet efficient framework is SimCLR [3], which uses augmented views in the current mini-batch as negative keys. On the other hand, MoCo [12] introduces an additional memory bank to maintain negative representations from neighboring mini-batches. It was found that contrastive learning can resist to semantic invariant corruptions due to its strong data augmentations [13].

Table 1: A short recapitulation of the related works.

	Related works	Major difference
Attack models	[10, 21, 31]	gradient-based attacks to yield feasible adversarial perturbations
	[2, 24]	optimization algorithms to find the "minimal" perturbations
Adversarial defenses	[7, 16, 21, 35, 26]	supervised learning using adversarial samples
	[4, 15, 17, 22]	unsupervised learning using adversarial samples
	[19, 34]	defenses that do not use adversarial examples
Contrastive learning	[12]	w/ memory banks
	[3, 25]	w/o memory banks

2.4. Adversarial self-supervised pre-training

Several recent works [4, 15, 22] began to treat the adversarial perturbation as a special data augmentation and adopt self-supervised learning to obtain the robust feature representations from data. The learned feature representations can further be transferred to various downstream tasks. However, unlike ordinary data augmentations, which are usually determined before training, adversarial examples are dynamically generated through training iterations and are greatly influenced by the current network parameters. So in order to capture the invariant features among them, very large batch sizes and long training epochs must be used. Note that each adversarial sample requires several rounds of forward propagation and backward gradient calculations. At this point, large batches and long training cycles will bring additional computational overhead far beyond ordinary contrastive learning.

To address the above problem, inspired by the work of [12], we develop a momentum-based adversarial pre-training framework, which consists of a standard clean memory bank and an additional adversarial memory bank. The newly proposed adversarial bank aims to alleviate the inconsistency of adversarial samples in each iteration, thereby reducing the learning difficulty and speeding up the training convergence.

3. Adversarial Momentum-Contrastive Learning

3.1. Preliminaries

We first recall the MoCo framework [12] for learning on clean data. Let C denote a set of data augmentation operations. For any augmentations c, c' in C, $(c(x), c'(x))$ forms a pair of *positive* augmented samples of x. The model includes a query encoder f_q and a key encoder f_k. Denote by $q = f_q(c(x))$ the query encoder representation, $k_+ = f_k(c'(x))$ the *positive key*, and k_- the *negative key*, which could be the representation of any other sample from the key encoder. The MoCo builds a memory bank $\mathcal{M}$ to keep the negative keys of recent mini-batches. Its optimization objective is then to minimize the following InfoNCE loss:

$$\begin{aligned} &\mathcal{L}_{\text{NCE}}(f_q(c(x)), f_k(c'(x)), \mathcal{M}) \\ &= -\log \frac{\exp(q \cdot k_+/T)}{\exp(q \cdot k_+/T) + \sum_{k_- \in \mathcal{M}} \exp(q \cdot k_-/T)}, \end{aligned} \tag{3.1}$$

where T is a temperature constant. During training, the mini-batch from the key encoder is subsequently enqueued into the memory bank $\mathcal{M}$ and at the same time the oldest ones are dequeued.

From an intuitive point of view, (3.1) forces the feature representation of each sample in the query encoder to be consistent with its augmented sample, and different from all other samples held in $\mathcal{M}$, thereby helping the query encoder learn the invariant feature representations among the data.

The query encoder f_q can be learned by standard back propagation, while the key encoder f_k is updated by the following rule [12]:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q, \tag{3.2}$$

where $m \in (0, 1)$ is a momentum coefficient, and θ_q and θ_k are parameters of query encoder and key encoder, respectively.

3.2. AMOC pre-training framework

Recent works [4, 15] considered adversarial perturbations as a special kind of data augmentations, and used contrastive learning technique to extract the robust feature representations in the data. However, adversarial examples are dynamically generated through training iterations and are greatly influenced by the current network parameters. In order to capture the invariant features among them, very large batch sizes and long training epochs must be used. Since the calculation of each adversarial example requires heavy computational resources, computational burden brought by large batches and long-term training can far exceed that in ordinary contrastive learning.

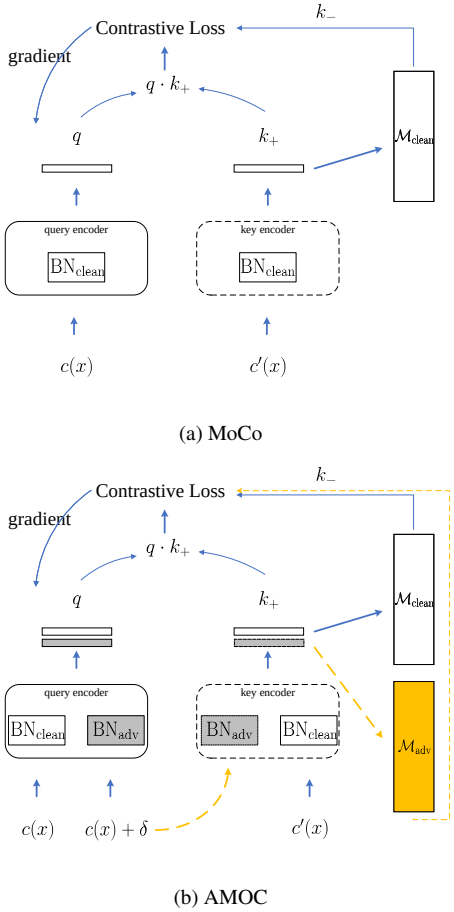

Figure 1: (a) MoCo [12] framework introduces the memory bank $\mathcal{M}_{\text{clean}}$ for historical information. (b) AMOC framework utilizes two memory banks $\mathcal{M}_{\text{clean}}$ and $\mathcal{M}_{\text{adv}}$ to maintain the historical clean and adversarial keys, respectively. Here c, c' denote distinct augmentations; δ is the perturbation injected into the query; $\text{BN}_{\text{clean}}, \text{BN}_{\text{adv}}$ are independent batch normalization modules for clean and adversarial samples; q, k_+ is a pair of encoded feature representations from the same sample; k_- denotes the negative key from the memory banks.

Inspired by the work of [12], we are to address the above problem by building two dynamic memory banks $\mathcal{M}_{\text{clean}}$ and $\mathcal{M}_{\text{adv}}$ to maintain clean and adversarial feature representations that are consistent across different mini-batches, respectively. As in [12], the clean memory bank provides negative keys from clean data for contrastive learning, thus guaranteeing the natural accuracy of the model. Whereas the newly proposed adversarial memory bank provides negative keys from historical adversarial samples, thereby alleviating the inconsistency of adversarial samples and ensuring a faster convergence.

Remark 1. *Aside from the same clean memory bank $\mathcal{M}_{\text{clean}}$ as in MoCo, AMOC introduces an additional adversarial memory bank $\mathcal{M}_{\text{adv}}$ to maintain adversarial keys. However, note that there are only clean keys in the original key encoder In order to update the adversarial memory bank, after each iteration, we feed the perturbed inputs of the query encoder to the key encoder (the dotted yellow lines in Figure 1), and then $\mathcal{M}_{adv}$ is updated in a first-in and first-out fashion, that is, the new adversarial keys are enqueued into $\mathcal{M}_{adv}$ while the oldest ones*

are dequeued.

In addition, since the distributions of adversarial and clean samples are different, we will utilize the dual Batch Normalization (BN) suggested in [33], i.e., one BN for clean samples and the other for adversarial ones. Then the corresponding encoder network is denoted as $f(\cdot;\mathrm{BN}_{\mathrm{clean}})$ or $f(\cdot;\mathrm{BN}_{\mathrm{adv}})$, respectively.

Now we give the formulation of the optimization objective of AMOC. Most of all, a robust neural network should guarantee the feature representations of clean samples, so the first part in our optimization objective is the standard MoCo loss [12]:

$$\mathcal{L}_{\mathrm{CCC}} = \mathcal{L}_{\mathrm{NCE}}(f_q(c(x);\mathrm{BN}_{\mathrm{clean}}), f_k(c'(x);\mathrm{BN}_{\mathrm{clean}}), \mathcal{M}_{\mathrm{clean}}),$$

where **CCC** means that **C**lean query encoder, **C**lean key encoder and **C**lean memory bank are used in this loss.

Furthermore, a robust neural network needs to be able to resist perturbations and learn invariant features under adversarial attacks. To this end, we formulate the second loss as follows

$$\mathcal{L}_{\mathrm{ACA}} = \mathcal{L}_{\mathrm{NCE}}(f_q(c(x)+\delta;\mathrm{BN}_{\mathrm{adv}}), f_k(c'(x);\mathrm{BN}_{\mathrm{clean}}), \mathcal{M}_{\mathrm{adv}}).$$

where **A**dversarial query encoder, **C**lean key encoder and **A**dversarial memory bank are used. Note that in the above loss, we add adversarial perturbation to the input of the query encoder, while leaving the input of the key encoder as it is. In this way, the query encoder is forced to learn common feature representations contained in adversarial and clean samples, i.e., robust features invariant under perturbations.

Combining the standard MoCo loss and the proposed adversarial loss, we arrive at our final optimization objective of the pre-training:

$$\mathcal{L} := \lambda\mathcal{L}_{\mathrm{CCC}} + (1-\lambda)\mathcal{L}_{\mathrm{ACA}}, \tag{3.3}$$

where the hyperparameter $\lambda \in (0,1)$ controls the effects of the clean samples and adversarial samples.

To minimize the objective (3.3), the parameter θ_q of the query encoder could be learned via standard back propagation, while the parameter θ_k should be updated by using (3.2), otherwise the key encoder will change rapidly and thus reduce the consistency of memory banks [12]. The entire optimization procedure of AMOC is summarized in Algorithm 1.

Algorithm 1 Training procedure of AMOC

Input: the image set $\{x\}$, augmentation operations C, the query encoder $f_q(\cdot;\theta_q)$ and the key encoder $f_k(\cdot;\theta_k)$.

1: **for** each mini-batch **do**
2: Sample augmentations c and c' from C and craft adversarial perturbations δ;
3: Compute three encoded feature representations, respectively:

$$f_q(c(x);\mathrm{BN}_{\mathrm{clean}}),\ f_q(c(x)+\delta;\mathrm{BN}_{\mathrm{adv}}),\ f_k(c'(x);\mathrm{BN}_{\mathrm{clean}});$$

4: Calculate the loss (3.3) and update the query encoder f_q via back propagation;
5: Update the key encoder f_k via (3.2);
6: Update the clean memory bank using $f_k(c'(x);\mathrm{BN}_{\mathrm{clean}})$ and adversarial memory bank using $f_k(c(x)+\delta;\mathrm{BN}_{\mathrm{adv}})$;

Output: the query encoder $f_q(\cdot;\mathrm{BN}_{\mathrm{adv}})$.

4. Experiment

4.1. Experimental setup

Datasets. In this section, we evaluate the performance of our approach on three commonly used datasets, CIFAR10 [18], CIFAR100 [18] and CIFAR10-C [13]. Both of CIFAR10 and CIFAR100 contain 60000 32×32 color images, which are divided into a training set of 50000 images and a test set of 10000 images, while CIFAR10-C consists of nineteen different types of semantic invariant corruptions. We use CIFAR10 and CIFAR100 to evaluate the adversarial robustness and CIFAR10-C to measure the persistence against common corruptions.

Attacks. To reliably evaluate the adversarial robustness of defense methods, several adversarial attacks including PGD [21], DeepFool [24], C&W [2], SLIDE [31] and AutoAttack [5] are adopted. All implementations of them are provided by FoolBox [27] except AutoAttack from the source code of [5]. The default settings of these attacks are listed in Table 2, wherein step size denotes the relative step size of PGD and SLIDE, the learning rate of C&W, and the overshoot of DeepFool, respectively. For brevity, denote by PGD20 the shorthand of PGD with 20 iterations.

Table 2: The basic setup for adversarial attacks in ℓ_∞, ℓ_1 and ℓ_2 norms.

	$\ell_\infty, \epsilon = 8/255$				$\ell_1, \epsilon = 12$		$\ell_2, \epsilon = 0.5$	
	PGD	PGD	DeepFool	AutoAttack	PGD	SLIDE	PGD	C&W
number of iterations	10	20	50	-	50	50	50	1000
step size	0.25	0.1	0.02	-	0.05	0.05	0.1	0.01

Baselines. The quality of learned feature representations is of most interest to pre-training methods, which can be evaluated by a linear classifier using the pre-trained feature representations. We choose adversarial contrastive learning (ACL) [15] as the baseline comparison, which is a state-of-the-art self-supervised framework without memory banks.

We also compare AMOC with several benchmark supervised defenses including PGD adversarial training (PGD-AT) [21], adversarial logits pairing (ALP) [16] and TRADES [35]. These defenses are implemented following the default settings of original papers. The adversarial samples required in the procedure of training are all crafted by PGD10 attack within $\epsilon = 8/255$.

Hyperparameters. There are three hyperparameters in our framework, including the momentum of m, the temperature of T and the length of K. We set $m = 0.999$ and $T = 0.2$ as suggested in [12] but use a smaller $K = 32768$ instead. The weight parameter in (3.3) is chosen as $\lambda = 0.5$ to balance the two loss terms (see Section 4.5 for sensitivity analysis).

Implementation details. We take the ResNet-18 [11] as the backbone and apply 5-step and 10-step l_∞ PGD attack [21] to generate adversarial perturbations for adversarial pre-training

and fine-tuning, respectively.

In the pre-training stage, the augmentations and projection head structure suggested in [3] are adopted. To optimize the loss function, we apply SGD (stochastic gradient decent) with weight decay of 5×10^{-4}, whose learning rate gradually climbs up to 0.1 by linear warmup for the first 10 epochs and then is decayed by the cosine decay schedule without restarts [20].

When it comes to fine-tuning on downstream classification tasks, we take the augmentations popularized by Residual Networks [11]: 4 pixels are reflection padded on each side, and a 32×32 crop is randomly sampled from the padded image or its horizontal flip. For full adversarial fine-tuning, we train the entire network using TRADES [35] for 100 epochs, just like ACL [15].

4.2. Pre-training performance

First, we evaluate the quality of the feature representations learned in pre-training. To this end, we add a linear classifier on top of the pre-trained query encoder network. The weights of the classifier are then determined by two ways, one using standard training and the other using adversarial training. These two ways are denoted as StdEv and AdEv, respectively. We compare AMOC with the recently proposed ACL [15].

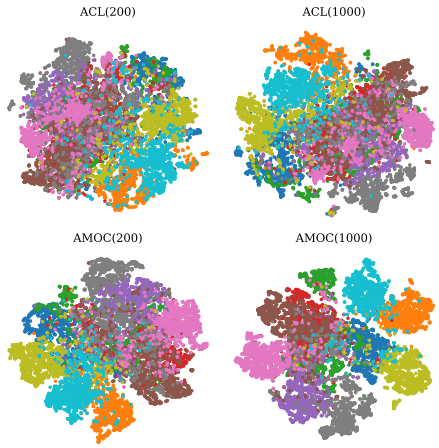

Figure 2: t-SNE [32] visualization of the feature representations of ACL [15] and AMOC on CIFAR-10 dataset after 200 and 1000 training epochs. Here ACL [15] takes a batch size of 512 while our method only uses a smaller batch size of 256. The numbers in ( ) represent the training epochs. The plot clearly shows that the feature distribution obtained by our approach has a better separability than that of ACL [15] under a smaller batch size and fewer training epochs.

The clean and robust accuracy are reported in Table 3 using different batch sizes and training epochs. As we can see that AMOC consistently outperforms ACL under the same configurations. In particular, the results of AMOC with a batch size of 64 have far outperformed those of ACL with a batch size of 512. Moreover, thanks to the constructed memory banks, our approach converges much faster than ACL. The performance of AMOC under AdEv using a batch size of 256 after 200 training epochs is comparable to that of ACL using a batch size of 512 after 1000 epochs. Moreover, it is obvious that MoCo has the best natural accuracy, but hardly any robustness. Compared to MoCo, our method achieves significant robustness improvements with a small loss of natural accuracy.

Table 3: Comparison of MoCo [12], ACL [15] and AMOC with different batch sizes and epochs on CIFAR10. **StdEv:** Train the classifier using *clean* samples through the frozen encoder. **AdEv:** Train the classifier using *adversarial* examples through the frozen encoder.

			StdEv		AdEv	
	Batch Size	Epochs	Clean	PGD20	Clean	PGD20
MoCo [12]	256	200	80.44	0.17	75.27	1.36
	256	1000	**90.91**	0.35	**87.00**	0.50
ACL [15]	64	200	70.53	26.24	62.83	35.76
	128	200	74.35	30.53	66.46	39.04
	256	200	77.14	32.72	68.60	40.36
	512	200	77.17	33.19	69.13	40.42
	512	1000	82.08	40.67	74.18	44.99
AMOC	64	200	77.74	37.56	73.29	43.39
	128	200	79.36	38.44	74.79	44.01
	256	200	79.10	37.12	74.36	43.36
	256	1000	86.12	**45.29**	81.91	**50.28**

Furthermore, we conduct the paired t-tests [28] between MoCo, AMOC and ACL based on 5 repetitive pre-training results after 200 epochs. We plot the p-values of statistical analysis under four measures in Figure 3. It is obvious that the performance differences are statistically significant under p-value < 0.05.

Next, we compare the running times of ACL and our approach in Table 4. It is observed that both methods take almost the same running time per epoch. But as reflected by the previous experimental results, the performance of AMOC after 200 training epochs is comparable to that of ACL after 1000 epochs, so the total running time of AMOC could be much less in practice.

Table 4: The running times of ACL [15] and AMOC with different batch sizes and epochs on CIFAR10. The times are evaluated on an Intel Xeon CPU E5-2680 v4 platform with 16 GB of memory and a single RTX 2080Ti GPU.

	Batch Size	Epochs	Total Time (hours)	Time per epoch (seconds)
ACL [15]	256	200	13.90	250.15
	512	200	13.40	241.15
	512	1000	67.01	241.15
AMOC	64	200	15.35	276.37
	256	200	13.33	239.94

Finally, we use the CIFAR10-C dataset [13] to further evaluate the robustness of the model against unseen semantic invariant corruptions. In Figure 4, we compare the classification accuracy of ACL and AMOC for each type of corruption. It is obvious that AMOC still outperforms ACL under different corruptions. Especially, under some corruptions like gaussian blur and impulse noise, the advantages of AMOC are particularly pronounced.

4.3. Comparison with supervised models after fine-tuning

In this section, we will demonstrate how the encoder network pre-trained by AMOC can further improve the downstream classification tasks. To this end, we first pre-train the encoder by AMOC, and then add a linear classifier and fine-tune the whole network using the adversarial loss of TRADES [35]. According to the results in the previous section, we choose a

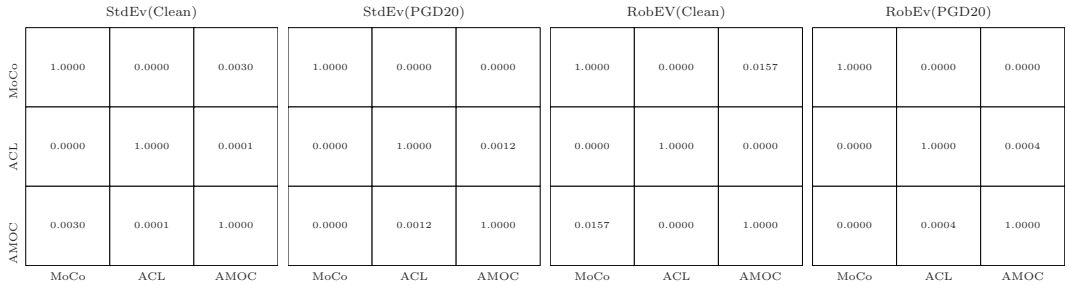

Figure 3: P-values of paired t-tests [28] between MoCo [12], ACL [15] and AMOC using 5 repetitive pre-training results. A lower p-value means a more significant difference in performance between the two methods.

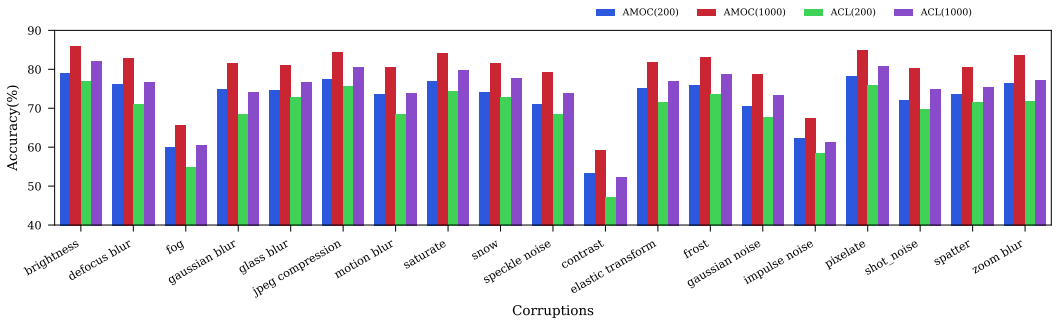

Figure 4: Comparision of ACL [15] and AMOC against unseen corruptions on CIFAR10-C. The numbers in ( ) represent the training epochs.

batch size of 256 and run 200 epochs in AMOC pre-training. We compared the fine-tuned results with several supervised defenses, including PGD-AT [21], ALP [16] and TRADES [35], on CIFAR10 and CIFAR100 datasets, respectively.

Table 5: Comparison of classification accuracy (%) between AT, ALP, TRADES and AMOC under various adversarial attacks on CIFAR10. We take **bold** type to indicate the best result, and underline type to indicate the second best result. AMOC: pre-trained results; AMOC†: fine-tuned results.

		$\ell_\infty, \epsilon = 8/255$			$\ell_1, \epsilon = 12$		$\ell_2, \epsilon = 0.5$	
	Clean	PGD20	DeepFool	AutoAttack	PGD50	SLIDE	PGD50	C&W
Standard Training	**95.28**	0.00	0.03	0.00	6.60	3.21	0.34	0.03
PGD-AT [21]	84.74	45.50	50.30	41.57	54.93	21.33	54.76	54.48
ALP [16]	85.07	49.09	**54.59**	44.13	57.55	23.03	57.19	56.32
TRADES [35]	81.11	51.89	51.89	47.07	58.20	27.84	59.53	57.17
AMOC	74.36	43.46	40.85	34.61	58.47	**31.92**	55.10	50.98
AMOC†	82.76	**53.51**	54.45	**48.75**	**59.85**	27.82	**60.57**	**58.41**

Table 6: Comparison of classification accuracy (%) between PGD-AT, ALP, TRADES and AMOC under various adversarial attacks on CIFAR100. We take **bold** type to indicate the best result, and underline type to indicate the second best result. AMOC: pre-trained results; AMOC†: fine-tuned results.

		$\ell_\infty, \epsilon = 8/255$			$\ell_1, \epsilon = 12$		$\ell_2, \epsilon = 0.5$	
	Clean	PGD20	DeepFool	AutoAttack	PGD50	SLIDE	PGD50	C&W
Standard Training	**76.34**	0.03	0.02	0.00	1.92	1.20	0.23	0.64
PGD-AT [21]	56.48	20.98	22.65	18.83	29.62	9.06	28.70	28.28
ALP [16]	55.71	25.97	25.32	22.02	32.93	13.18	32.38	30.21
TRADES [35]	54.95	26.72	24.39	21.70	35.47	**15.84**	34.26	31.24
AMOC	42.11	20.50	14.95	11.90	31.27	14.78	28.57	22.87
AMOC†	57.64	**28.90**	**26.88**	**24.08**	**36.66**	15.64	**35.71**	**32.78**

It can be found in Table 5 that AMOC with fine-tuning (AMOC†) surpasses other defenses under most attacks. Although PGD-AT and ALP can achieve better natural accuracy, they perform poorly under AutoAttack, 7% and 4% less than ours. Recall that AMOC employs the same adversarial loss as TRADES for fine-tuning, but AMOC outperforms TRADES by 1.5% regardless of natural accuracy or robustness. It indicates that the encoder pre-trained using AMOC is beneficial to improve the robustness while preserving high natural accuracy.

Similar trends can also be observed on CIFAR100 as shown in Table 6. AMOC is not only robust against most attacks, but also outperforms other defense methods on natural accuracy. Therefore, adversarial training on top of the pre-trained encoders is arguably a good choice for efficiency and robustness purposes.

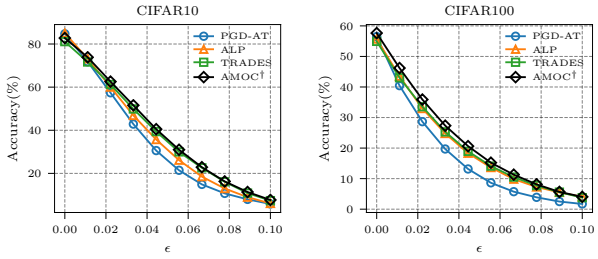

Figure 5: Comparison between PGD-AT, ALP, TRADES and AMOC under PGD20 attacks with various perturbation budgets ϵ.

Although all defense methods are adversarially trained within $\epsilon = 8/255$, it would be better if they could extrapolate to lager perturbations. Figure 5 depicts the relationship between classification accuracy and increasing perturbation budgets under PGD20 attack. Compared with PGD-AT and ALP, AMOC maintains stable and extraordinary robustness, which demonstrates the reliability and scalability of our approach.

4.4. Other variants

Note that in the second loss term $\mathcal{L}_{\text{ACA}}$ of (3.3), we inject **A**dversarial perturbations into the query encoder, using the key encoding of **C**lean augmented data to provide the positive key and the **A**dversarial memory bank to provide the negative keys. If we replace the loss term $\mathcal{L}_{\text{ACA}}$ with $\mathcal{L}_{\text{ACC}} := \mathcal{L}_{\text{NCE}}(f_q(c(x) + \delta; \text{BN}_{\text{adv}}), f_k(c'(x); \text{BN}_{\text{clean}}), \mathcal{M}_{\text{clean}})$, then we can derive another pre-training version whose optimization objective is $\lambda\mathcal{L}_{\text{CCC}} + (1-\lambda)\mathcal{L}_{\text{ACC}}$. For simplicity, we call this version ACC. It differs from original AMOC (ACA) in that only a single clean memory bank is used to provide negative keys. However, as can be seen from the first part of Table 8, although this version shows good robustness and natural accuracy, its performance is worse than that of ACA, which demonstrates that the newly proposed adversarial memory is indeed essential to improve the performance of the model further.

Some people might consider feeding adversarial samples to the key encoder. However, this is not a good choice because it will yield vague positive keys, making the model difficult to learn. We provide the corresponding verification in the second part of Table 8. It is observed that injecting perturbations into the key encoder can greatly deteriorate the performance of the model.

Table 7: Configurations of different variants of AMOC. ✓ means injecting the adversarial perturbations to the corresponding encoder while ✗ means not injecting the perturbations.

	ACA	ACC	AAA	AAC	CAA	CAC
Query encoder	✓	✓	✓	✓	✗	✗
Key encoder	✗	✗	✓	✓	✓	✓
Memory bank	$\mathcal{M}_{\text{adv}}$	$\mathcal{M}_{\text{clean}}$	$\mathcal{M}_{\text{adv}}$	$\mathcal{M}_{\text{clean}}$	$\mathcal{M}_{\text{adv}}$	$\mathcal{M}_{\text{clean}}$

Table 8: Comparison of classification accuracy (%) of possible variants on CIFAR10. **StdEv:** Train the classifier using clean samples with frozen encoder. **AdEv:** Train the classifier using adversarial examples with frozen encoder.

	StdEv		AdEv	
	Clean	PGD20	Clean	PGD20
ACA	79.10	**37.12**	**74.36**	**43.36**
ACC	78.74	36.87	74.08	42.89
AAC	10.00	10.00	10.00	10.00
AAA	68.52	34.44	63.66	39.20
CAC	78.46	0.00	24.82	15.53
CAA	**79.29**	0.00	27.12	15.08

4.5. Sensitivity analysis

The length of the memory banks decides the amount of the historical information used in each training epoch. We investigate its influence by considering the robustness and natural accuracy under various length choices. As shown in Figure 6, the performance of the our approach is not sensitive to the length of the memory banks, and both robustness and natural accuracy are very stable as K changes.

The balance hyperparameter λ in (3.3) controls the effects of the adversarial samples and clean samples. We plot the natural accuracy and robustness with respect to different values of λ in Figure 7. It is observed that either too large or too small λ can deteriorate the performance of the model, which means that the clean loss $\mathcal{L}_{\text{CCC}}$ and adversarial loss $\mathcal{L}_{\text{ACA}}$ are equally important in our framework. Moreover, a too large λ seems more damaging to robustness, which implies that overemphasizing clean samples can have a larger negative impact.

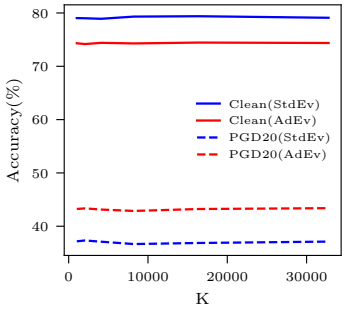

Figure 6: The performance of AMOC across different length of memory banks. **StdEv:** Train the classifier using clean samples with frozen encoder. **AdEv:** Train the classifier using adversarial examples with frozen encoder.

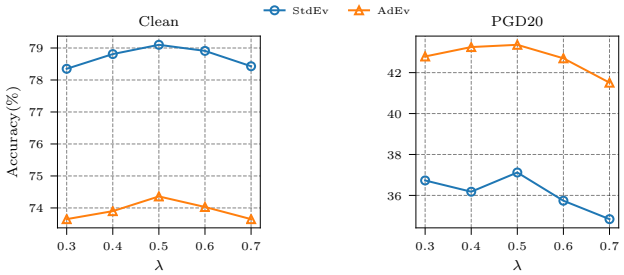

Figure 7: The clean accuracy and PGD20 robustness of AMOC with respect to different balance hyperparameter λ. **StdEv:** Train the classifier using clean samples with frozen encoder. **AdEv:** Train the classifier using adversarial examples with frozen encoder.

5. Conclusion

In this work, we incorporate adversarial perturbation into the perspective of data augmentation, and propose a simple but efficient momentum-based contrastive pre-training to improve the robustness and natural accuracy of the neural networks. Thanks to the designed memory banks, compared to the previous adversarial self-supervised learning approach, the proposed model can learn more discriminative feature representations using a smaller batch size and far fewer epochs. However, since we still rely on adversarial attacks to yield augmented samples in the training, the approach still bears a heavy computational burden. How to further accelerate the pre-training is a promising direction in the future.

References

[1] Bui A., Le T., Zhao H., Montague P., Camtepe S., & Phung, D. Understanding and achieving efficient robustness with adversarial supervised contrastive learning. *arXiv preprint arXiv:2101.10027*, 2021.

[2] Carlini N. & Wagner D. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

[3] Chen T., Kornblith S., Norouzi M. & Hinton G. E. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, 2020.

[4] Chen T., Liu S., Chang S., Cheng Y., Amini L. & Wang Z. Adversarial robustness: from self-supervised pre-training to fine-tuning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[5] Croce F. & Hein Matthias. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning (ICML)*, 2020.

[6] Deng J., Dong W., Socher R., Li L., Li K., & Fei-Fei L. ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[7] Dong C., Liu L. & Shang J. Double descent in adversarial training: an implicit label noise perspective. In *International Conference on Learning Representations (ICLR)*, 2022.

[8] Engstrom L., Tsipras D., Schmidt L. & Madry A. A rotation and a translation suffice: fooling cnns with simple transformations. In *International Conference on Machine Learning (ICML)*, 2019.

[9] Geirhos R., Rubisch P., Michaelis C., Bethge M., Wichmann F. & Brendl W. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. In *International Conference on Learning Representations (ICLR)*, 2019.

[10] Goodfellow I. J., Shlens J. & Szegedy C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[11] He K., Zhang X., Ren S. & Jian S. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[12] He K., Fan H., Wu Y., Xie S.& Girshick R. Momentum contrast for unsupervised visual representation learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[13] Hendrycks D. & Dietterich T. Benchmarking neural network robustness to common corruptions and perturbations. In *International Conference on Learning Representations (ICLR)*, 2019.

[14] Huang H., Wang Y., Erfani S., Gu Q., Bailey J. & Ma X. Exploring architectural ingredients of adversarially robust deep neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2021

[15] Jiang Z., Chen T., Chen T. & Wang Z. Robust pre-training by adversarial contrastive learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[16] Kannan H., Kurakin A. & Goodfellow I. J. Adversarial logit pairing. arXiv preprint arXiv:1803.06373, 2018.

[17] Kim M., Tack J. & Hwang S. J. Adversarial self-supervised contrastive learning. In *Advances in Neural Information Processing Systems (NIPS)*, 2020.

[18] Krizhevsky A. & Hinton G. E. Learning multiple layers of features from tiny images. *Techinical report, Citeseer*, 2009.

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

[20] Loshchilov I. & Hutter F. Sgdr: stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

[21] Madry A., Makelov A., Schmidt L. & Tsipras D. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018.

[22] Meng Z., Dong Y., Sachan M., & Wattenhofer, R. Self-supervised contrastive learning with adversarial perturbations for robust pretrained language models. *arXiv preprint arXiv:2107.07610*, 2021.

[23] Mikołajczyk A. & Grochowski M. Data augmentation for improving deep learning in image classification problem. In *International Interdisciplinary PhD Workshop (IIPhDW)*, 2018.

[24] Moosavi-Dezfooli S., Fawzi A. & Frossard P. DeepFool: a simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[25] Oord A., Li Y. & Vinyals. Representation learning with contrastive predictive coding. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[26] Rade R. and Moosavi-Dezfooli S. Helper-based adversarial training: reducing excessive margin to achieve a better accuracy vs. robustness trade-off. In *International Conference on Machine Learning (ICML)*, 2021

[27] Rauber J., Brendel W. & Bethge M. Foolbox v0.8.0: A Python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017

[28] Rice J. A. Mathematical statistics and data analysis. *Cengage Learning*, 2006.

[29] Rusak E., Schott L., Zimmermann R., Bitterwolf J., Bringmann O., Bethge M. & Brendel W. A simple way to make neural networks robust against diverse image corruptions. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[30] Szegedy C., Zaremba W., Sutskever I., Bruna J., Erhan D. Goodfellow I. J. & Fergus R. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014.

[31] Tramèr F. & Boneh D. Adversarial training and robustness for multiple perturbations. In *Advances in Neural Information Processing Systems (NIPS)*, 2019.

[32] Van der Maaten L. & Hinton G. E. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 2008.

[33] Xie C., Tian M., Gong B., Wang J., Yuille A. & Le Q. Adversarial examples improve image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[34] Xu C., Li X., & Yang M. An orthogonal classifier for improving the adversarial robustness of neural networks, *Information Sciences*, 591(2022), 251–262.

[35] Zhang H., Yu Y., Jiao J., Xing E., Ghaoui L. & Jordan M. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning (ICML)*, 2019.