Cluster-Guided Adversarial Perturbations for Robust Contrastive Learning Supplementary Material

Seongyun Seo Sungmin Han Jeonghyun Lee Sangkyun Lee*
School of Cybersecurity, Korea University
Seoul, South Korea
{tjtjddus111,sungmin_15,nomar0107,sangkyun}@korea.ac.kr

A. Algorithm for cluster-guided adversarial training

Algorithm 1 Stage 2: Cluster-guided adversarial training

Input: Dataset $\mathbb{D}$, pre-trained encoder f_{θ_1}, number of training steps T.
Output: Adversarially robust encoder f_{θ_2}.
Initialize f_{θ_2} with pre-trained weights of f_{θ_1}.
Repeat for each epoch:
Extract representation vectors of $\mathbb{D}$ using f_{θ_2}.
Apply spherical k-means clustering to partition the representation vectors.
Compute cluster centroids $\{C_k\}_{k=1}^{K}$.
Determine a threshold τ based on the distances between cluster centroids.
for step $= 0, \ldots, T-1$ **do**
  Sample a mini-batch $B = \{x_0, \ldots, x_{m-1}\}$.
  for all $x_i \in B$ (in parallel) **do**
    $z_i \leftarrow f_{\theta_1}(x_i)$.
    $C_{x_i} \leftarrow$ centroid of the cluster assigned to x_i.
    $C_{\text{dis}(x_i)} \leftarrow$ centroid of the cluster farther than τ from x_i.
    $x_i^{\text{adv}} \leftarrow x_i + \delta_i^*$, where δ_i^* solves Eq. (8) in the main paper.
  end for
  Update θ_2 by minimizing $\sum_{i=0}^{m-1} \mathcal{L}_{\text{CGACL}}(x_i; \theta_2)$.
end for

B. Experimental details

B.1. Procedure of the proof-of-concept experiment

We conduct a proof-of-concept experiment to evaluate the adversarial effectiveness of our perturbation generation strategy. The experiment proceeds in five steps: **(i) pretraining the encoder:** we train an encoder using SimCLR [3] to learn a structured representation space; **(ii) clustering representations:** we apply spherical k-means clustering to the representations extracted from the encoder trained in step (i); each cluster is expected to consist of semantically similar samples; the number of clusters k is chosen based on the optimal silhouette score; **(iii) generating perturbations:** for each sample, we generate a perturbation that moves it away from the centroid of its assigned cluster and toward the centroid of the n-th nearest neighboring cluster ($n \geq 2$); **(iv) training the linear classifier:** to evaluate the adversarial effectiveness of the generated perturbations, we attach a linear classifier to the frozen encoder from step (i) and train it using labeled data; **(v) measuring attack success rates:** we evaluate the attack success rates of the perturbations using the classifier trained in step (iv).

For this experiment, we employ the ResNet-18 [7] architecture as the encoder. CGACL generates perturbations by optimizing Eq. (8) in Section 3.2 using a 5-step PGD with a perturbation budget of $\epsilon = 8/255$. In contrast, MaxCL and DeACL skip step (ii) and directly generate perturbations in step (iii): MaxCL does so by maximizing the contrastive loss, while DeACL does so by minimizing the cosine similarity with representations extracted from the encoder in step (i). On the other hand, Supervised skips (ii)–(iii) and after step (iv) directly generates perturbations by maximizing the cross-entropy loss with respect to ground-truth labels. All other steps are identical across methods. To ensure precise evaluation, we measure the attack success rates only on samples correctly classified by the model trained in step (iv). As shown in Fig. 1 in Section 3.1, when n is sufficiently large, our approach achieves higher attack success rates than than existing methods, demonstrating its superior adversarial effectiveness.

B.2. Implementation of CGACL

Stage 1: Standard contrastive learning We implement SimCLR using solo-learn [4], a library of self-supervised methods. Following [3], we train models for 1000 epochs. We use a batch size of 512 and a learning rate of 1.0 for CIFAR-10, CIFAR-100, and STL-10, and a batch size of

*Corresponding author

Table 1. Mean±standard deviation of evaluation results under standard linear fine-tuning. The best and second-best results are marked with boldface and underline, respectively. † indicates results taken directly from the original paper, for which standard deviations are unavailable. 'n/a' denotes missing values due to the absence of required information.

Method	CIFAR-10			CIFAR-100		
	SA	RA	AA	SA	RA	AA
ADVCL	80.85†	50.45†	42.57†	48.34†	27.67†	19.78†
DeACL	80.17†	<u>53.95</u>†	45.31†	52.79†	<u>30.74</u>†	<u>20.34</u>†
DYNACL	75.38±0.04	48.48±0.03	45.56±0.02	45.85±0.02	22.76±0.06	19.43±0.07
DYNACL++	79.79±0.27	49.37±0.48	47.05±0.45	**53.93**±0.20	21.97±0.10	19.71±0.11
DYNACL-AIR++	<u>81.86</u>±0.10	49.01±0.23	<u>47.12</u>±0.21	<u>53.90</u>±0.41	23.20±0.16	20.69±0.14
CGACL (ours)	**83.19**±0.10	**54.09**±0.07	**50.09**±0.01	51.56±0.12	**31.34**±0.05	**23.12**±0.07
Method	STL-10			ImageNet-100		
	SA	RA	AA	SA	RA	AA
ADVCL	n/a	n/a	n/a	n/a	n/a	n/a
DeACL	<u>80.87</u>±0.86	<u>65.86</u>±0.87	<u>58.50</u>±0.69	48.28±0.08	<u>27.16</u>±0.17	17.40±0.13
DYNACL	69.61±0.05	48.76±0.14	47.17±0.08	55.06±0.07	16.26±0.01	11.18±0.02
DYNACL++	70.97±0.04	49.91±0.67	47.80±0.71	<u>56.24</u>±0.07	22.66±0.05	<u>19.42</u>±0.01
DYNACL-AIR++	71.36±0.25	50.23±0.30	48.42±0.24	47.08±0.11	21.64±0.09	18.92±0.07
CGACL (ours)	**84.68**±0.11	**69.53**±0.22	**66.09**±0.30	**56.66**±0.14	**31.91**±0.07	**23.90**±0.34

256 with a learning rate of 0.5 for ImageNet-100. We adopt the LARS optimizer with a momentum of 0.9 and apply cosine learning rate decay with a 10-epoch linear warm-up. The weight decay is set to 1e-4 for CIFAR-10 and CIFAR-100, and 1e-5 for STL-10 and ImageNet-100. The temperature parameter for contrastive loss is fixed at 0.5. For data augmentation, we apply the random resized crop (with crop size 32 for CIFAR-10 and CIFAR-100, 96 for STL-10, and 224 for ImageNet-100) using a scale range between 0.08 and 1.0. Additional augmentations include color jitter (strength 0.5 for CIFAR-10 and CIFAR-100, and 1.0 for STL-10 and ImageNet-100) with a probability of 0.8, grayscale with a probability of 0.2, Gaussian blur with a probability of 0.5, and horizontal flip with a probability of 0.5.

Stage 2: Cluster-guided adversarial training We implement spherical k-mean clustering using kmcuda [1] and AFK-MC^2 [2] to improve training efficiency and scalability. kmcuda provides GPU-accelerated k-means clustering, while AFK-MC^2 is a fast and simple seeding algorithm for k-means clustering. The number of clusters, k, is determined based on the silhouette score. The optimal k values across different random seeds and datasets are as follows: for CIFAR-10, the values are 70, 60, 70, 90, and 75; for CIFAR-100, 150, 200, 100, 165, and 130; for STL-10, 215, 140, 165, 155, and 175; and for ImageNet-100, 110, 100, and 110. In the case of STL-10, the training dataset consists of $5,000$ labeled data and $100,000$ unlabeled data, where the unlabeled set includes additional object categories (e.g., bears, rabbits, trains, buses) not present in the labeled set. This explains why the optimal number of clusters for STL-10 is considerably larger than its number of classes (10).

For cluster-guided adversarial training, the model is trained for 200 epochs on CIFAR-10, 100 epochs on CIFAR-100, 1000 epochs on STL-10, and 100 epochs on ImageNet-100. We use a batch size of 256 and a learning rate of 0.5 for CIFAR-10, CIFAR-100, and STL-10. For ImageNet-100, we instead set the batch size to 128 and the learning rate to 0.25. For data augmentation, we apply weak augmentation during training. Specifically, for CIFAR-10, CIFAR-100, and STL-10, we apply random cropping after padding to the original image size (32 for CIFAR-10 and CIFAR-100, and 96 for STL-10), followed by horizontal flipping with a probability of 0.5. For ImageNet-100, images are first resized so that their shorter side is 256 pixels with aspect ratio preserved, then center-cropped to 224×224, and finally horizontally flipped with a probability of 0.5. The hyperparameter λ is set to 6. Additionally, we adopt the dual batch normalization method from [8], which maintains separate batch normalization layers for clean and adversarial examples.

B.3. Fine-tuning Settings

Standard linear fine-tuning We train the linear classifier for 25 epochs using clean examples, keeping the encoder pre-trained with CGACL fixed. Training is performed with a batch size of 512, using SGD with a momentum of 0.9 and a weight decay of 2e-4. The initial learning rate is set to 0.1 for CIFAR-10, 10.0 for CIFAR-100 and STL-10, and

Table 2. Performance comparison between CGACL and DeACL, the most robust state-of-the-art method, across different model architectures. Evaluation is conducted using standard linear fine-tuning. Results are reported as standard accuracy (SA), robust accuracy (RA), and Auto-Attack accuracy (AA), all presented in percentages (%). The best results are marked with boldface.

Architecture	Method	CIFAR-10			CIFAR-100		
		SA	RA	AA	SA	RA	AA
MobileNetV2	DeACL	79.32	**51.36**	44.56	39.51	24.57	16.24
	CGACL (ours)	**81.73**	**51.36**	**47.17**	**44.82**	**26.39**	**18.88**
ResNet-18	DeACL	80.17	53.95	45.31	**52.79**	30.74	20.34
	CGACL (ours)	**83.19**	**54.09**	**50.09**	51.56	**31.34**	**23.12**
ResNet-34	DeACL	84.99	54.72	49.60	51.15	**31.64**	22.89
	CGACL (ours)	**85.92**	**54.90**	**51.90**	**54.31**	31.57	**24.75**
VGG16	DeACL	85.18	51.06	47.36	52.23	28.78	20.80
	CGACL (ours)	**85.34**	**51.56**	**48.36**	**53.71**	**28.87**	**22.16**
Inception-v2	DeACL	84.25	53.79	48.82	49.69	31.59	22.41
	CGACL (ours)	**84.74**	**54.06**	**50.36**	**52.61**	**31.65**	**24.03**
ViT-Small	DeACL	**82.09**	34.76	33.59	47.08	17.69	14.73
	CGACL (ours)	77.60	**38.52**	**35.71**	**49.54**	**19.23**	**16.96**
Architecture	Method	STL-10			ImageNet-100		
		SA	RA	AA	SA	RA	AA
MobileNetV2	DeACL	**73.84**	54.83	43.33	39.18	24.06	14.56
	CGACL (ours)	71.91	**56.36**	**46.47**	**43.16**	**28.26**	**19.68**
ResNet-18	DeACL	80.87	65.86	58.50	48.28	27.16	17.40
	CGACL (ours)	**84.68**	**69.53**	**66.09**	**56.66**	**31.91**	**23.90**
ResNet-34	DeACL	83.72	71.97	66.31	49.76	27.28	19.26
	CGACL (ours)	**87.22**	**75.41**	**71.74**	**59.18**	**32.26**	**25.76**
VGG16	DeACL	86.25	72.68	65.06	43.78	28.30	17.02
	CGACL (ours)	**88.69**	**75.85**	**70.06**	**52.54**	**33.50**	**23.58**
Inception-v2	DeACL	82.71	67.68	60.66	45.90	26.98	18.52
	CGACL (ours)	**84.40**	**69.56**	**66.07**	**56.14**	**33.76**	**26.36**
ViT-Small	DeACL	86.33	70.55	67.83	**56.36**	14.60	10.44
	CGACL (ours)	**89.05**	**78.72**	**77.91**	56.24	**23.22**	**19.52**

1.0 for ImageNet-100, with a decay factor of 10 at epochs 15 and 20. To complement the SLF results in the main paper, Table 1 reports both the mean and standard deviation of accuracy. This provides a more complete view of the stability and consistency of each method. Standard deviations are unavailable for methods whose results are taken from the original papers and are marked as †. 'n/a' indicate cases that could not be obtained due to missing components.

Adversarial linear fine-tuning We train the linear classifier for 25 epochs using adversarial examples generated by a 10-step ℓ_∞-PGD attack with $\epsilon = 8/255$ and $\alpha = 2/255$. The batch size is set to 512, and the learning rate to 0.1 for CIFAR-10. We use the SGD optimizer with the same momentum, weight decay, and learning rate decay settings as in standard linear fine-tuning.

Adversarial full fine-tuning Unlike standard and adversarial linear fine-tuning, which use cross-entropy loss and keep the encoder fixed, adversarial full fine-tuning optimizes both the encoder and the linear classifier using TRADES loss. The batch size is reduced to 128, and the learning rate is set to 0.001 for CIFAR-10. All other hyperparameter settings, including optimizer and learning rate decay, are identical to those in adversarial linear fine-tuning.

C. Additional experiments

C.1. Impact of model architectures

Table 2 compares the performance of our proposed method with DeACL, adopted as the baseline due to its strong robustness among state-of-the-art methods, across six model architectures: MobileNetV2 [11], ResNet-18 [7], ResNet-34, VGG16 [12], Inception-v2 [13], and ViT-Small [5]. This evaluation demonstrates the generalizability of our method,

Table 3. Comparison of total training time (in hours) across adversarial contrastive learning methods on CIFAR-10, CIFAR-100, STL-10, and ImageNet-100. The fastest and second-fastest methods are highlighted in boldface and underline, respectively.

Method	CIFAR-10 (hours)	CIFAR-100 (hours)	STL-10 (hours)	ImageNet-100 (hours)
ADVCL	85.78	86.29	143.95	476.47
DeACL	**9.34**	**9.23**	**52.92**	**204.85**
DYNACL	52.49	52.33	110.83	240.69
DYNACL++	53.68	53.74	113.23	254.05
DYNACL-AIR++	72.64	73.87	129.56	328.03
CGACL (ours)	<u>12.93</u>	<u>10.45</u>	<u>77.93</u>	<u>215.66</u>

Table 4. Breakdown of total training time (in hours) for CGACL into key components.

Component	CIFAR-10 (hours)	CIFAR-100 (hours)	STL-10 (hours)	ImageNet-100 (hours)
Contrastive learning	7.36	7.23	32.98	190.94
k-means clustering	1.18	1.01	13.38	8.42
Adversarial training	4.38	2.21	31.57	16.29
Total training time	12.93	10.45	77.93	215.66

CGACL, across architectures with varying capacity.

Across all settings, our method achieves stronger robustness than DeACL. A minor exception occurs with ResNet-34 on CIFAR-100, where DeACL slightly surpasses CGACL in robust accuracy by 0.07%. However, this gap is negligible and not observed in other cases. Moreover, in the same setting, CGACL achieves notably higher standard accuracy (+3.16%), leading to stronger overall performance. Consequently, experimental results confirm that CGACL generalizes well across diverse architectures, from lightweight to deep models.

C.2. Comparison of computational efficiency

We compare the computational cost of our proposed method, CGACL, against existing adversarial contrastive learning methods. As shown in Tab. 3, CGACL achieves the second-fastest training time across all datasets. Although DeACL is slightly faster, this comes at the cost of noticeably lower generalization and robustness, as demonstrated in earlier evaluations. These results suggest that CGACL balances computational efficiency and robustness without requiring substantial overhead. Prior work [15] further demonstrates that two-staged adversarial contrastive learning methods train faster than joint methods [6, 9, 14] that perform adversarial training and contrastive learning simultaneously. In line with this finding, DeACL and our method, both adopting a two-stage design, achieve the fastest and second-fastest training times, respectively.

To provide further insight, Table 4 presents a detailed breakdown of the total training time of CGACL. As shown, the majority of the time is spent on standard model training (contrastive learning and adversarial training), while clustering (including silhouette score computation and cluster selection) introduce only minor overhead. Specifically, the clustering-related steps account for only 9.13%, 9.67%, 17.17%, 3.90% of the total training time on CIFAR-10, CIFAR-100, STL-10, and ImageNet-100, respectively. All experiments were conducted using a single NVIDIA GeForce RTX 2080 Ti GPU.

Table 5. Cluster purity measured on representations learned by supervised learning (Supervised) and contrastive learning (Contrastive) across four datasets. Higher values represent that clustering learned representations yields semantically consistent clusters.

	Supervised	Contrastive
CIFAR-10	0.91	0.81
CIFAR-100	0.71	0.43
STL-10	0.72	0.76
ImageNet-100	0.62	0.57

C.3. Visualization of minimum perturbations

Figure 1 visualizes the minimal adversarial perturbations required to cause misclassification in each SLF-trained model, across different adversarial contrastive learning methods. The perturbations are computed on test samples from CIFAR-10, CIFAR-100, STL-10, and ImageNet-100.

The results show that models trained with our method require consistently larger perturbations to be misclassified, indicating stronger robustness. For certain datasets (*e.g.*, STL-10 or ImageNet-100), ADVCL results are omitted due to the unavailability of its implementation under those settings, which are indicated as 'n/a' in the figure.

Table 6. Impact of the threshold parameter τ on performance. τ is set as the mean of pairwise cosine distances between centroids scaled by $\{0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$.

τ	CIFAR-10			CIFAR-100			STL-10			ImageNet-100		
	SA	RA	AA	SA	RA	AA	SA	RA	AA	SA	RA	AA
0.25	82.64	53.67	49.58	51.35	32.39	23.86	85.24	70.64	67.11	56.62	33.12	25.38
0.5	82.93	54.49	50.36	51.69	32.23	24.05	84.70	70.15	66.53	57.02	32.78	24.72
0.75	83.19	54.44	50.59	51.66	31.52	23.56	84.55	70.41	66.84	56.68	32.20	24.26
1.0 (ours)	83.19	54.09	50.09	51.56	31.34	23.12	84.68	69.53	66.09	56.66	31.91	23.90
1.25	83.50	53.25	49.16	51.71	31.04	22.91	84.23	70.03	66.28	56.90	31.44	23.98
1.5	83.41	53.28	49.29	51.43	31.29	23.15	84.45	70.39	67.29	57.18	32.44	23.92
DeACL (baseline)	80.17	53.95	45.31	52.79	30.74	20.34	80.87	65.86	58.50	48.28	27.16	17.40

C.4. Impact of the number of clusters

Figure 2 shows the performance of CGACL across varying number of clusters k on CIFAR-10, CIFAR-100, STL-10, and ImageNet-100. The results demonstrate that our k-means clustering-based adversarial contrastive learning method is insensitive to the choice of k, provided it exceeds a certain threshold. Notably, the k values selected by the silhouette score always lie above this threshold. Moreover, the accuracy at the k value selected by the silhouette score closely matches the best performance in each case, validating the effectiveness of our heuristic for determining k.

C.5. Analysis of cluster purity

Table 5 reports the cluster purity [10] obtained by clustering representations learned through supervised learning and contrastive learning (SimCLR). Cluster purity is defined as the proportion of the most frequent class within each cluster, and the final score is averaged over all clusters. The value ranges from 0 to 1, where higher values indicate that clusters are predominantly composed of samples from a single class. To compute this metric, we applied k-means clustering to the learned representations with the number of clusters k determined by the silhouette score, and then measured the cluster purity on the resulting clusters. As shown in Tab. 5, contrastive learning achieves cluster purity comparable to supervised learning, indicating that it induces representation spaces where semantically similar samples are located nearby. In particular, contrastive learning attains higher cluster purity on STL-10, which contains 100,000 unlabeled samples but only 5,000 labeled ones, highlighting its ability to exploit large-scale unlabeled data. In conclusion, contrastive learning produces clusters that are semantically coherent.

C.6. Impact of the threshold parameter τ

We assessed the impact of the threshold parameter τ used to select target clusters. To this end, we measured performance under varying values of τ on CIFAR-10, CIFAR-100, STL-10, and ImageNet-100, where τ was set by scaling the mean of pairwise cosine distance between centroids by $\{0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$.

As shown in Tab. 6, only minor differences appear in SA, RA, and AA as τ varies, indicating that the overall performance of our method is stable with respect to τ. This stability comes from the clustering structure formed by contrastive learning. As shown in Sec. C.5, contrastive representations tend to form semantically coherent clusters, meaning that even nearby clusters are likely to be semantically dissimilar. Consequently, perturbations targeting nearby clusters can still induce adversarial effects, making our methods insensitive to the choice of τ. Moreover, our method achieves higher robust accuracy than the strong baseline DeACL across nearly all τ values.

References

[1] K-means and K-nn using NVIDIA CUDA, 2019. https://github.com/src-d/kmcuda. 2

[2] Olivier Bachem, Mario Lucic, S. Hamed Hassani, and Andreas Krause. Fast and provably good seedings for k-means. In *NeurIPS*, pages 55–63, 2016. 2

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

[4] 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*, 23(56):1–6, 2022. 1

[5] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3

[6] Lijie Fan, Sijia Liu, Pin-Yu Chen, Gaoyuan Zhang, and Chuang Gan. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? In *NeurIPS*, pages 21480–21492, 2021. 4

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 3

[8] Ziyu Jiang, Tianlong Chen, Ting Chen, and Zhangyang Wang. Robust pre-training by adversarial contrastive learning. In *NeurIPS*, pages 16199–16210, 2020. 2

[9] Rundong Luo, Yifei Wang, and Yisen Wang. Rethinking the effect of data augmentation in adversarial contrastive learning. In *ICLR*, 2023. 4

[10] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. 5

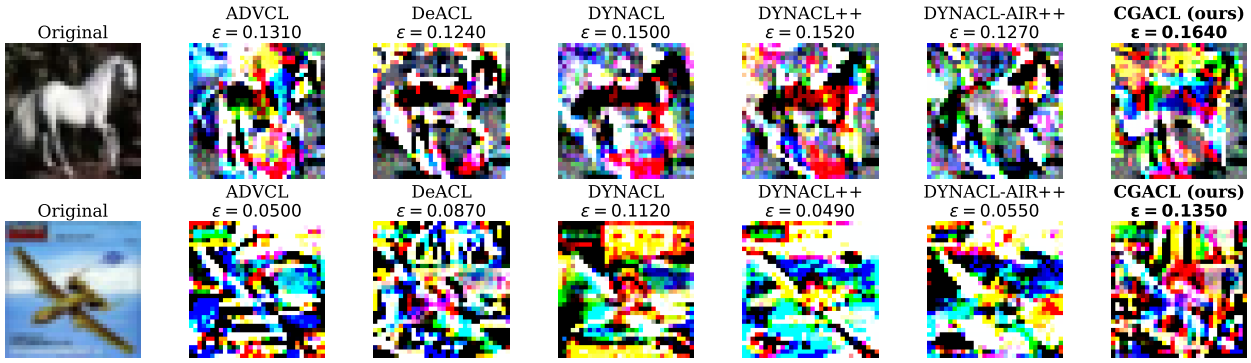
[11] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018. 3

[12] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 3

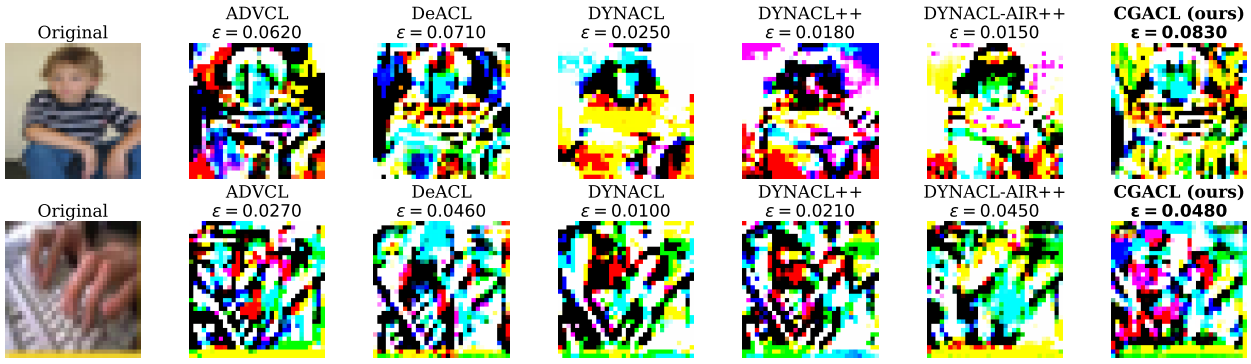
[13] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 3

[14] Xilie Xu, Jingfeng Zhang, Feng Liu, Masashi Sugiyama, and Mohan Kankanhalli. Enhancing adversarial contrastive learning via adversarial invariant regularization. In *NeurIPS*, pages 16783–16803, 2023. 4

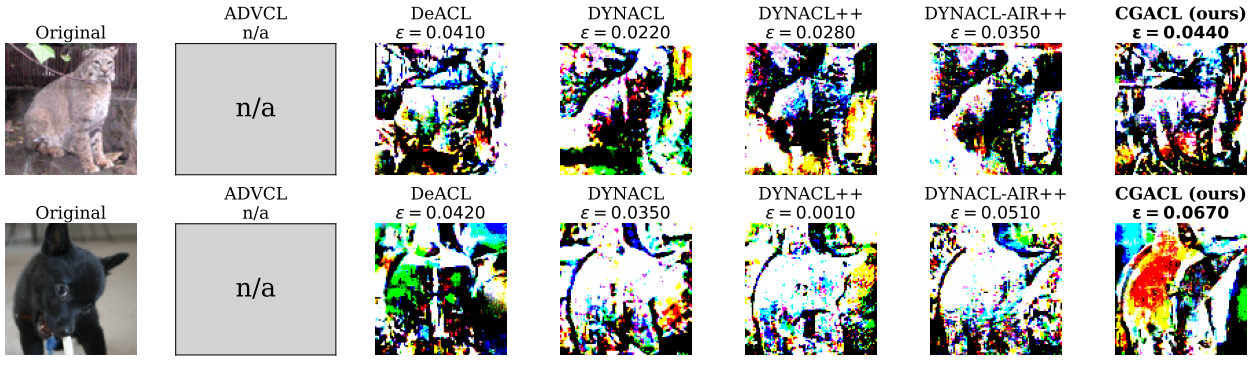
[15] Chaoning Zhang, Kang Zhang, Chenshuang Zhang, Axi Niu, Jiu Feng, Chang D. Yoo, and In So Kweon. Decoupled adversarial contrastive learning for self-supervised adversarial robustness. In *ECCV*, pages 725–742, 2022. 4



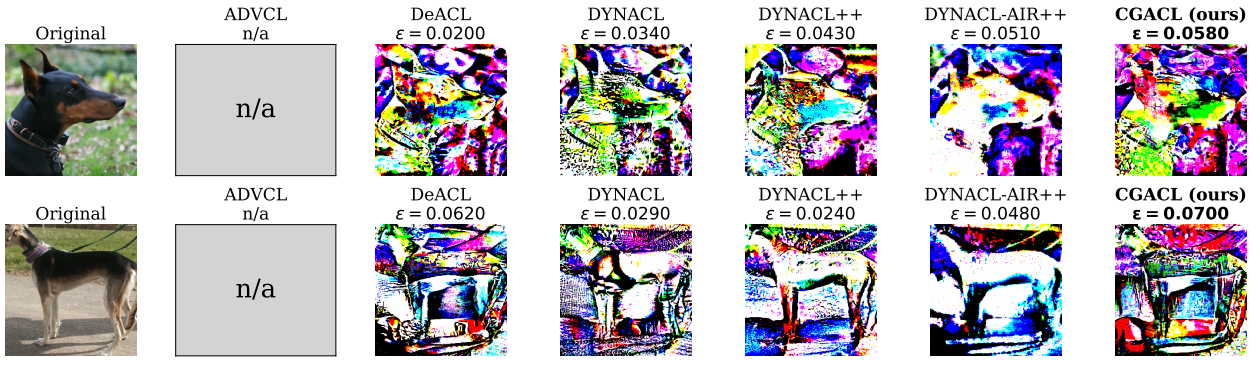
(a) CIFAR-10



(b) CIFAR-100

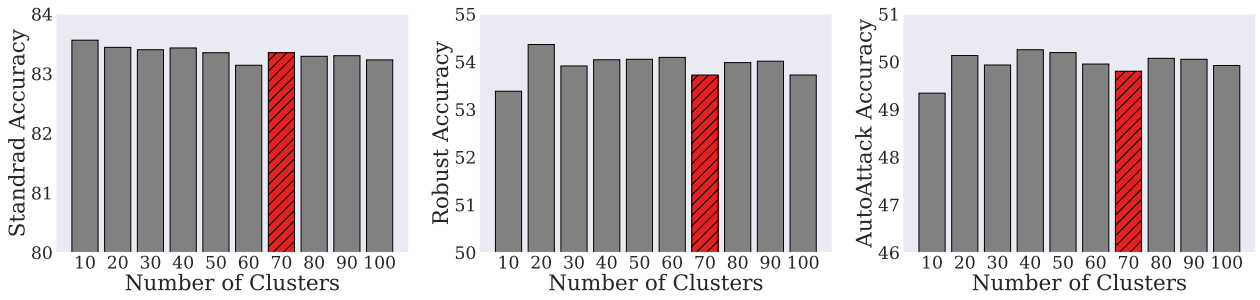


(c) STL-10

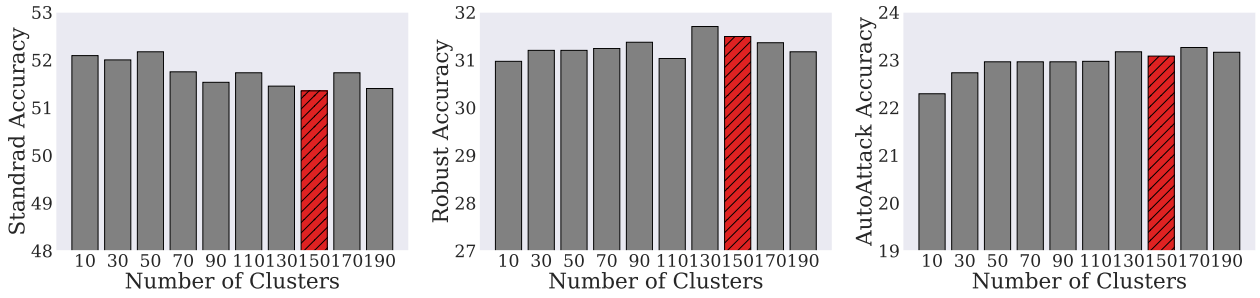


(d) ImageNet-100

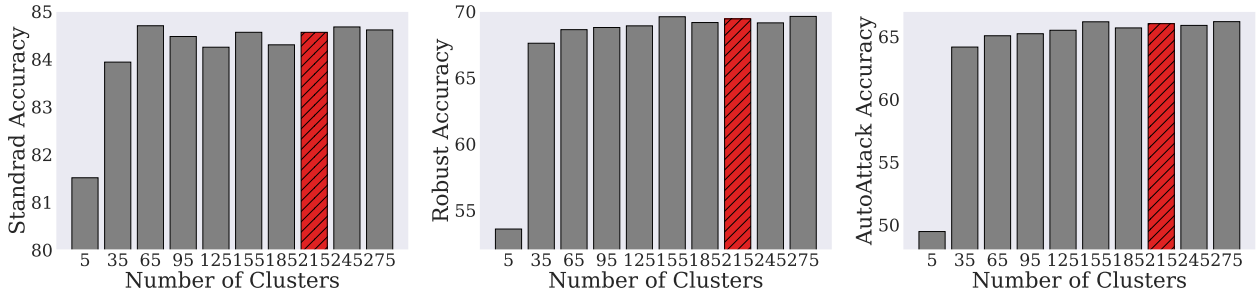
Figure 1. Visualization of minimum perturbations needed to fool each model, where the perturbation magnitude is denoted as ϵ and measured using the ℓ_∞-norm. Models trained with CGACL demonstrate higher robustness by requiring larger perturbations to cause misclassification. ADVCL results are excluded (marked as 'n/a') for STL-10 and ImageNet-100 due to the unavailable implementations.



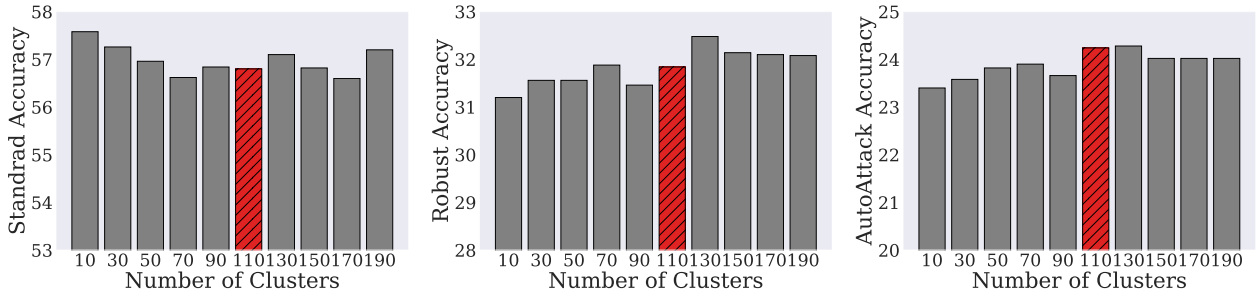
(a) CIFAR-10



(b) CIFAR-100



(c) STL-10



(d) ImageNet-100

Figure 2. Performance of CGACL across different number of cluster k. The accuracy at the k with the highest silhouette score is highlighted in the red stripes (results are shown for the random seed set to 0).