A *Light* Recipe to Train Robust Vision Transformers

Anonymous Author

***Abstract*—In this paper, we ask whether Vision Transformers (ViTs) can serve as an underlying architecture for improving the adversarial robustness of machine learning models against evasion attacks. While earlier works have focused on improving Convolutional Neural Networks, we show that also ViTs are highly suitable for adversarial training to achieve competitive performance. We achieve this objective using a custom adversarial training recipe, discovered using rigorous ablation studies on a subset of the ImageNet dataset. The canonical training recipe for ViTs recommends strong data augmentation, in part to compensate for the lack of vision inductive bias of attention modules, when compared to convolutions. We show that this recipe achieves suboptimal performance when used for adversarial training. In contrast, we find that omitting all heavy data augmentation, and adding some additional bag-of-tricks (ε-warmup and larger weight decay), significantly boosts the performance of robust ViTs. We show that our recipe generalizes to different classes of ViT architectures and large-scale models on full ImageNet-1k. Additionally, investigating the reasons for the robustness of our models, we show that it is easier to generate strong attacks during training when using our recipe and that this leads to better robustness at test time. Finally, we further study one consequence of adversarial training by proposing a way to quantify the semantic nature of adversarial perturbations and highlight its correlation with the robustness of the model. Overall, we recommend that the community should avoid translating the canonical training recipes in ViTs to robust training and rethink common training choices in the context of adversarial training.**

***Index Terms*—Adversarial Robustness, Adversarial Training, Computer Vision, Vision Transformer**

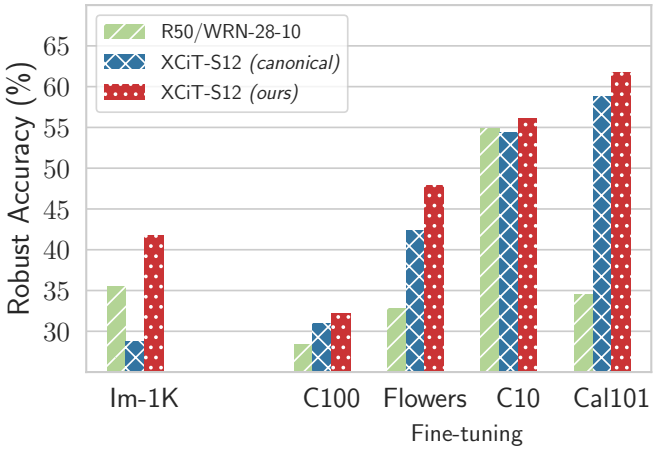


Fig. 1: **A *light* recipe is better!** Comparison of our proposed recipe with light data augmentation and the canonical one for ViTs (XCiT-S12 in this experiment) in pre-training on ImageNet-1k and finetuning on four different datasets. Our recipe boosts robust accuracy by 13.1% on ImageNet-1k and up to 5.5% on downstream finetuning tasks. For further comparison, we also include ResNet-50 (with smooth activation function [1, 2] – GELU [3]). Since a WideResNet-28-10 performs better for low-resolution datasets, we use it as a baseline for C10 and C100. Abbreviations for datasets are 1) *Im-1k*: ImageNet-1k, 2) *C100*: CIFAR-100, 3) *Flowers*: Oxford-flowers, 4) *C10*: CIFAR-10, and 5) *Ca101*: Caltech-101.

I. Introduction

Adversarial training [4], which has emerged as one of the most successful defenses against evasion attacks targeting machine learning models [5, 6], consists of training against worst-case inputs to make networks robust. However, it incurs a large generalization gap [7–9], and closing this gap is a key challenge. Existing work that tackles this problem can be broadly classified into three categories: 1) improvements in the adversarial training *mechanism* [4, 10–12], 2) modifications to the *training data* [13–15], and 3) modifications of the *neural network architecture* for robust training [1, 16]. Our work focuses on the latter category, by improving robustness with new architectures while improving the training recipe.

Vision Transformers for adversarial training. Several earlier works have innovated on activation functions [1, 2, 16] for CNNs or their structure [17] to improve robustness. While such advancements in CNNs are helpful and CNNs are the *de facto* standard architecture for adversarial training, we show that a drastic boost in adversarial robustness can be achieved by switching the architecture class itself, i.e., by using Vision Transformers (ViTs) [18]. In fact, even non-adversarially trained ViTs show some signs of higher robustness than conventional CNNs when considering non-adversarial perturbations [2, 19–21], which further encourages their use in robust training. Inspired by these features shown by ViTs, prior work [2, 22, 23] has attempted to train ViTs with adversarial training but failed to replace CNNs as the dominant architecture on robustness benchmarks [24]. Since ViTs are competitive with CNNs in standard training [25, 26], it is imperative to ask whether their suboptimal performance in adversarial training is fundamental or simply an artifact of sub-par training recipes. With a systematic investigation, we identify that the latter is true and propose a simple, yet highly effective recipe to boost the performance of ViTs in adversarial training.

Canonical ViTs training recipes are suboptimal for adversarial training. Due to the lack of strong inductive bias in ViTs, they need custom training recipes [27], which include heavy data augmentation, to achieve optimal performance. We observe that using these canonical recipes with adversarial training leads to sub-optimal performance, even suggesting that state-of-the-art ViTs [28] fail to significantly outperform

conventional CNN architectures, such as ResNets [29]. We show that optimizing the training recipe boosts the performance of ViTs from suboptimal to *state-of-the-art* (Fig. 1).

Improved training recipe (weaker data augmentation is better!). Instead of just taking the canonical training recipe used by DeiT [30] and successive works [28, 31–33], we first analyze some ViT variations and architectural components that could make ViTs more suitable for adversarial training. We then identify a set of important parameters that have a fundamental role in adversarial training, such as adversarial training warm-up, data augmentations, and weight decay. We go beyond the canonical training recipes commonly used for ViT-like models by doing a thorough search for the optimal values of these parameters: we observe that the optimal choices for non-adversarial training drastically differ from those for adversarial training. In fact, we find that while the use of strong data augmentation is recommended for standard training [27, 30], this is detrimental to adversarial training: not only does using light data augmentation improve the robustness of ViTs, but it does so without sacrificing the clean accuracy. Our recommendation to use weaker data augmentation is also *surprising* since earlier works have argued for stronger data augmentation in adversarial training [2], albeit with a warm-up in the augmentation intensity, as well as the same weight decay as the one used for standard training.

Generalization of our recipe across scales, architectures, and datasets. Improving over the canonical training recipe requires a rigorous ablation study, which has a high computational cost on large-scale datasets and models.[1] To circumvent this challenge, we optimize our recipe with a subset of the ImageNet-1k dataset and smaller models. However, for widespread usage, it is imperative that the benefits of our recipe generalize across diverse scenarios. We first show that the proposed recipe outperforms the canonical one at scale, i.e., on the full ImageNet-1k dataset. Next, we show that not only it performs better across the XCiT transformer class [28], but also for other transformers (such as DeiT [30] and PoolFormer [32]), as well as for modern CNNs (ConvNeXt [34]). Finally, we show that the gains of our recipe in pre-training on ImageNet-1k also directly transfer when finetuning on smaller datasets of both high- and low- resolution.

Delving deeper into adversarial robustness of ViTs. We conduct two analyses to better understand why our recipe brings an improvement: attack effectiveness and semantic nature quantification. Since large-scale adversarial training uses only few-step attacks, its performance depends on the strength of adversarial examples generated with few-step attacks. We uncover that, throughout the whole training duration, few-step attacks are more effective for a ViT trained with our recipe than for a ViT trained with the canonical recipe, as well as more effective than a conventional CNN architecture. This results in a model which is overall more robust to strong attacks at test time. Next, we propose a new way to quantify the semantic nature of adversarial perturbations [35] for robust ViTs. We show, with a quantitative method, that the adversarial perturbations targeting XCiT-S12 have more semantic features than those targeting a GELU ResNet-50.

Key contributions. We make the following key contributions:

- Through a rigorous ablation study, we uncover a light yet effective adversarial training recipe for ViTs. In particular, we find that the use of weak data augmentation, in contrast to strong augmentation in the canonical recipe, achieves *state-of-the-art* performance on the ImageNet-1k dataset (compared with the other models on RobustBench [24]), with models having up to 47.60% AutoAttack accuracy and 73.76% clean accuracy.
- We further show that our proposed recipe generalizes across different scales of datasets and models, and different classes of ViT architectures.
- We demonstrate that the advantage of our recipe in pre-training also leads to benefits when finetuning on downstream datasets. Across four low- and high-resolution datasets, our recipe achieves up to 5.5% higher robust accuracy than the canonical recipe.
- We identify that the high robustness of ViTs is also related to the effectiveness of adversarial attacks on them. Simultaneously, we quantify that adversarial perturbations targeting robust ViTs have semantic characteristics.

Paper Outline. We provide a brief overview of adversarial training and the ViT architecture in Section II. In Section III, we first identify the limitations of the canonical recipe and then conduct an ablation study for the proposed training recipe. In Section IV, we first show the advantage of our recipe on the full-scale ImageNet-1k dataset across different architectures. We later demonstrate its downstream benefits in finetuning. In Section V, we examine why ViTs are highly successful in adversarial training. In Section VI, we present the related works and conclude the paper with a discussion in Section VII.

Open-sourced artifacts. Finally, we share the code on GitHub[2] to enable the community both to reproduce our results and fine-tune the models on further datasets. In the same repository, we also share the checkpoints of our models for five datasets, numerous architectures, and perturbations to enable other researchers to fine-tune the models, as well as run further analyses.

II. Background

Adversarial training [4] has shown to be an effective and reliable method to defend against adversarial examples, and most of the subsequent work about defenses against adversarial examples is based on it. On the other hand, ViTs are an emerging class of architectures that achieve competitive performance on standard computer vision benchmarks [25]. In this section, we give an overview of both.

[1]Doing such ablation on full ImageNet-1k with XCiT-S would take 2632 TPUv3 hours (i.e., 179 TPUv3 days).

[2]https://github.com/dedeswim/vits-robustness-torch

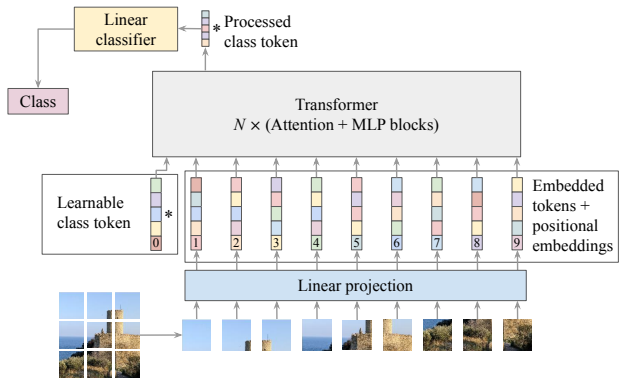


Fig. 2: **Overview of the transformer architecture.** The different phases of a vision transformer model: 1) patchification 2) addition of the positional encodings and of the `[cls]` token 3) processing through N transformer blocks 4) classification. Further details about ViTs and some variants are in Appendix A. Image reproduced from Dosovitskiy *et al.* [18].

A. Overview of adversarial training

Adversarial training [4] is one of the most successful defenses against adversarial examples. It consists of generating adversarial examples at each step and training the model using the generated perturbed data instead of the original, clean data. Given a model with parameters $\boldsymbol{\theta}$, input data $\mathbf{x}$ with label y sampled from a distribution p_{data}, a set of allowed inputs $\mathbb{S}$ and a loss $\mathcal{L}$, adversarial training formally consists of optimizing the following min-max problem:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}} \left[\max_{\boldsymbol{\delta}\in\mathbb{S}} \mathcal{L}(\mathbf{x}+\boldsymbol{\delta}, \mathrm{y}; \boldsymbol{\theta}) \right]. \quad (1)$$

Several further techniques have been proposed to improve adversarial training and the trade-off [10] between robustness and accuracy. Some of them include the optimization of a different objective [10, 36], early stopping to avoid the so-called robust overfitting [37], tuned data augmentation [13], weight averaging [13] etc.

Metrics in adversarial training. It is common to use the following two metrics to measure performance in adversarial training: a) *Clean accuracy*: the accuracy of the model evaluated on the original data. b) *Robust accuracy.*: the accuracy of the model on a dataset of adversarial examples generated from the original data using an attack. To avoid overestimating the robustness of our networks, it is critical to evaluate the models using a strong attack when generating adversarial examples. We employ AutoAttack [38] for the robust accuracy evaluation. AutoAttack is considered the de-facto standardized method to assess the adversarial robustness of classification models. It is an ensemble of four different parameter-free attacks, three white- and one black-box. We provide more details about the evaluation procedure of our experiments in Section III.

B. Overview of Vision Transformers

The Transformer architecture [39] is an architecture for sequence transduction and Natural Language Processing (NLP) tasks (e.g., machine translation) based on the attention mechanism [40]. This architecture includes a series of so-called *Multi-Head Attention* layers, each followed by a *Multi-Layer Perceptron* block. Every layer has a residual connection. This architecture takes as input a series of words, which are tokenized and embedded into vectors of size d_{model}. The Transformer architecture can also be easily adapted for computer vision tasks [18]. The resulting architecture is called *Vision Transformer* (ViT). In particular, it is possible to divide the input images into non-overlapping patches, which are embedded into tokens, and then fed to the transformer. We give additional in-depth details about the various components of this architecture, as well as more information about the ViT variants used in this work in Appendix A.

III. Finding an effective adversarial training recipe for vision transformers

We now aim at finding an effective training recipe to leverage the potential that ViTs have shown for standard training. We first highlight the limitations of the canonical standard training recipes which use strong data augmentation. We then conduct an ablation study of major design choices: 1) a warm-up for the perturbation budget, 2) data augmentation policy, and 3) weight decay. As a result, we show the factor which influences performance the most: surprisingly, strong data augmentation leads to sub-par performance.

Setup. We validate the ablations in this section with a simplified setup for the sake of efficiency: we train thirty-two models on a subset of 100 random classes of ImageNet-1k and validate them using APGD-CE [38], the first attack of the well-established AutoAttack [38]. However, in Section IV we will validate the findings of this section on the full ImageNet-1k dataset [41], using the full ensemble of AutoAttack. Throughout the paper, we mainly focus on untargeted ℓ_∞ attacks, and for ImageNet-100 and ImageNet-1k on perturbations of magnitude $4/255$, as this is the most studied scenario so far [24] (with the exception of Section IV-A, where we also test our recipe on the ℓ_2 threat model with $\varepsilon = 3.0$). We evaluate the checkpoint which has the highest FGSM accuracy throughout the training, to prevent robust overfitting [37]. Additional details about the training setup and procedure are in Appendix D.

A. The canonical training recipe leads to suboptimal performance in adversarial training

While early attempts with ViTs succeeded only with large-scale pre-training, a lot of effort has been put to achieve good performance without the need of pre-training them on very large datasets. In particular, both Touvron *et al.* [30] and Steiner *et al.* [27] observe that strong data augmentation techniques are needed, i.e., MixUp, CutMix, RandAugment, and Random Erasing, as they help compensate for the lack of a strong vision prior such as convolutions. Similarly, other work on ResNets and CNNs, in general, observed that stronger data augmentation improves the generalization of standard training [34, 42].

Canonical training recipe. We refer to *canonical training recipe* as the one used in the original XCiT paper [28], which,

in turn, is borrowed from DeiT's paper [30], and used for the large majority of ViT variations (e.g., CaiT [31] and PoolFormer [32]), and modern CNNs (e.g., ConvNeXt [34]). We summarize this recipe below. When adversarially training XCiT-S, a ViT variant, if we use the canonical recipe, we observe very poor performance when compared to the equivalent state-of-the-art ResNet-50 with GELU activation function by Bai *et al.* [2]. They use a setup that does not differ from the standard setup for ResNets and achieve almost one fourth better robust accuracy (35.51% vs. 28.70%). Given this difference, considering that on standard training XCiT-S performs better than ResNet-50, it is natural to investigate whether a better setup can lead to stronger results. In this section, we rigorously analyze the design space of adversarial training and propose a better training recipe. In Table I, we show a summary of why the canonical recipe is sub-optimal for XCiT-S, with further, rigorous results in Table II, which we discuss in the next sections.

TABLE I: **Limitation of canonical training recipe.** While borrowing the canonical recipe in adversarial training largely succeeds for CNNs, i.e., ResNet-50, it leads to suboptimal performance in ViTs, i.e., XCiT-S. Directly translating the canonical recipe to adversarial training gives a *false* impression that ViTs are not suitable for adversarial training.

Architecture	Standard training	Adversarial training
	Clean accuracy	AutoAttack accuracy
ResNet-50	76.0 [42, 43] [3]	**35.51** [2]
XCiT-S	**82.0** [28]	28.70

Canonical training recipe

- Strong data augmentations (MixUp + CutMix + RandAugment + Random Erasing)
- Small weight decay (0.05 using AdamW on ImageNet-1k)

B. Architecture choice: Architectural innovations significantly benefit adversarial training

Our objective is to understand how innovations in ViTs architecture, post the conception in Dosovitskiy *et al.* [18], directly benefit adversarial training. In particular, we focus on DeiT [30], CaiT [31], and XCiT [28]. These architectures are a natural choice, as each improves upon the latter. CaiT improves over DeiT by introducing *Class Attention* while XCiT further improves CaiT using *Cross-Covariance Attention*. We provide detailed descriptions of each architecture in Appendix B. Our results in Table IIa show that Class Attention in CaiT helps with the fit to adversarial training, and Cross-Covariance Attention boosts, even more, the performance. For this reason, we choose XCiT as the base architecture for our experiments. To speed up the ablation study, we use a smaller variant of XCiT, XCiT-N12.

[3]For ResNet we show the clean accuracy of the model available in the torchvision library, as reported by Wightman *et al.* [42]. However, this model is trained without heavy augmentation, similarly to the adversarially trained ResNet-50 by Bai *et al.* [2], even though Wightman *et al.* [42] show that heavy data augmentation benefits ResNets as well.

C. Data-augmentation: Adversarial training of ViTs requires weak augmentation

The success of ViTs strongly depends on the use of heavy data augmentation and appropriate regularization (Steiner *et al.* [27] and Touvron *et al.* [30]), often attributed to their lack of inductive bias for vision tasks. Simultaneously, adversarial training for CNNs also benefits from heavy data augmentation [13], albeit on low-resolution datasets, thus it is natural to start using heavy data augmentation in adversarial training of ViTs. However, we find this choice highly sub-optimal, as *weaker* data augmentation achieves much better performance with adversarial training (Table IIc). We run a thorough ablation, considering all sixteen combinations of the four key data augmentation policies: CutMix [44], RandAugment [45], MixUp [46], and Random Erasing [47]. We give an in-depth description and examples of the various data augmentation techniques in Appendix C. In all the training runs, we always apply basic augmentations such as horizontal flipping, random resize-rescale, and color jitter. We show the top seven setups in Table IIc, ranked by APGD-CE accuracy (full results in Table IX in Appendix E). Surprisingly, the augmentation setup that leads to the best results in terms of APGD-CE accuracy is the one with no additional augmentations, apart from the basic ones listed above, together with the one that uses only Random Erasing. These setups improve the robust accuracy by 3.84% over the canonical strategy of using heavy data augmentation. This phenomenon is likely arising due to the inherent regularization imposed by adversarial training, where strong adversarial perturbations already make the optimization much harder thus leading to better performance without heavy augmentation. Moreover, we note that the models with the best robust accuracy are also the ones with the best clean accuracy, suggesting that using only light data augmentation does not affect clean accuracy. We note that the setup with only Random Erasing has the same robust accuracy as the one with no heavy augmentation but has a slightly larger clean accuracy. Despite this, we choose as a setup for the next experiments the one without Random Erasing, to keep the overall setup as simple as possible. We will validate this choice in Section IV-A.

D. Optimization setup: Tuning attack curriculum and additional regularization brings further improvements

Epsilon warm-up. Bai *et al.* [2] observe that adversarially training a DeiT on the full ImageNet-1k dataset would fail using the same setup as the one in the DeiT paper. For this reason, we attempt training an XCiT-N12 to see if the training succeeds. Even though the training run succeeds, the model struggles in the first few epochs. A possible solution could be to use the following attack curriculum: we make the task easier for the first few epochs and then gradually make it harder. We warm up the adversarial perturbation budget (ε) by linearly increasing ε for a fixed number of warm-up epochs.

TABLE II: **Beyond canonical choices: identifying best adversarial training setup for ViTs.** We analyze choices in architecture, data-augmentation, and optimization setup to identify the best training setup for adversarial training. We use the ImageNet-100 dataset and measure the robust accuracy using the APGD-CE attack. Our analysis uncovers intriguing trends: one is a counter-intuitive phenomenon where *weaker* data augmentations lead to better performance in ViTs with adversarial training.

(a) **Comparison of different ViT-like architectures.** Innovation in ViTs architectures has a relevant impact on adversarial training performance. We observe that the cross-covariance attention-based XCiT architecture achieves significantly better performance than the others.

Architecture	Parameters	GFLOPs	Accuracy	
			Clean	*APGD-CE*
DeiT-S [30]	22M	4.61	62.52	33.32
CaiT-S-12 [31]	25M	4.76	70.20	35.84
XCiT-N12 [28]	3M	0.56	48.46	30.48
XCiT-S12 [28]	26M	4.81	**85.06**	**54.80**

(b) **Attack curriculum.** We warm up ε by linearly increasing it for a fixed number of early epochs. It benefits both clean and robust accuracy. (Arch: XCiT-N12).

Epochs	Accuracy	
	Clean	*APGD-CE*
0	48.46	30.48
5	52.04	32.86
10	54.62	33.84
20	56.10	**34.88**
30	**56.12**	34.54

(c) **Weak data augmentation is better.** The strategies that perform best are those with just Random Erasing or no heavy augmentation at all. We report the seven best results, and the baseline recipe with all data augmentations, in this table (full results in Table IX in Appendix E). In all the runs in this table we keep weak data augmentations that are commonly used (random flip and crop, and color jitter). (Arch: XCiT-N12)

Data Augmentation Policy				Accuracy	
MixUp	CutMix	RandAugment	Random Erasing	*Clean*	*APGD-CE*
✗	✗	✗	✓	**67.28**	**39.22**
✗	✗	✗	✗	66.78	**39.22**
✓	✗	✗	✗	61.04	38.56
✓	✗	✗	✓	60.46	38.26
✓	✓	✗	✗	62.04	38.18
✗	✗	✓	✗	65.34	37.64
✗	✗	✓	✓	64.76	37.62
✓	✓	✓	✓	56.64	35.38

(d) **Large weight decay helps.** The best results are obtained with 0.5 weight decay, which is $10\times$ larger than the 0.05 weight decay used in the canonical recipe. (Arch: XCiT-N12)

Weight Decay	Accuracy	
	Clean	*APGD-CE*
0	66.44	39.02
0.001	66.40	39.04
0.01	66.28	38.66
0.05	67.16	39.30
0.1	67.28	39.92
0.5	**68.78**	**42.02**
1.0	67.68	40.88

Our results (Table IIb) show that using 20 epochs as warm-up duration gives a significant increase in both clean accuracy and APGD-CE accuracy. This suggests that, while gradually increasing the difficulty of the task does not prove useful for models with larger inductive bias, such as ResNets [48], it can help for models that have a low inductive bias (e.g., ViTs).

Weight decay. As pointed out by Pang *et al.* [48], weight decay has an important role to make models more robust: a larger weight decay helps reduce the generalization gap for robust accuracy. For this reason, we ablate several values of weight decay, in different orders of magnitude. The weight decay used to train XCiT originally was 0.05 [28], but we get the best results with the weight decay equal to 0.5 (Table IId), both in terms of clean and robust accuracy. We hypothesize that the regularization introduced by the larger weight decay helps as we have removed heavy data augmentation.

IV. Validating our training recipe at scale

We now test the best setup found in the previous section on the full ImageNet-1k dataset, with a range of architectures and model sizes. We show that not only our recipe achieves strong results for XCiT-S when compared to the canonical recipe, but it also enables better results on other modern architectures, and other model sizes in the case of XCiT. Finally, we show that we can also pre-train and fine-tune transformers using this recipe for a larger attack budget: XCiTs pre-trained in this way can be robustly fine-tuned to a diverse set of downstream datasets achieving competitive performance.

Our training recipe. Based on our previous findings, we use a) a 10-epochs, linear warm-up for ε, b) better-tuned data augmentation, by using just weak data augmentation (i.e., random resize, crop and horizontal flipping, and color jitter), and c) 0.5 weight decay. We do a 10 epochs warm-up, instead of 20 (as what Table IIb would suggest), because ImageNet-1k is 10 times larger than ImageNet-100: after 10 epochs the model will have seen enough *easy* images with small perturbations, to then continue the training with the full perturbation budget.

Proposed training recipe

- 10-epoch Linear ε-warmup
- Only basic data augmentation (random-resize-and-crop + horizontal-flipping + color-jitter).
- High weight decay (0.5 using AdamW on ImageNet-1k)

TABLE III: **Weak data augmentation with large weight decay is better than heavy data augmentation curriculum.** In adversarial training of ViTs, Bai *et al.* [2] previously recommended the use of strong data augmentation, but with a progressive curriculum. We observe that using *only* weak data augmentations strategies, together with larger weight decay, brings better results than using progressive strong data augmentation increased in the first epochs. For a fair comparison, we use identical network architecture and training setup as Bai *et al.* [2]. The model marked with †, trained by Bai *et al.* [2] with the canonical recipe, failed the training, while our implementation of the same model is marked with *.

Model	Accuracy	
	Clean	AutoAttack
GELU ResNet-50 [2] *(c)*	67.38	35.53
DeiT-S† [2] *(c)*	—	—
DeiT-S [2] *(c + heavy augmentation curriculum)*	66.50	35.50
DeiT-S* *(c, ours)*	66.30	32.70
DeiT-S *(ours)*	66.80	37.90
XCiT-S12 *(ours)*	72.34	41.78

A. *Validating success on the full ImageNet-1k dataset*

Intuition of why the recipe should scale up. We observe that the data augmentation setup, as well as weight decay, could interact with the dataset size: a smaller dataset may need stronger data augmentation, as the model could overfit. In particular, this was experimentally validated for ViTs by Steiner *et al.* [27] and Touvron *et al.* [30]. On the other side, we note that we use *minimal* data augmentation on the smaller ImageNet-100. Hence, we argue that, as ImageNet-1k is larger than ImageNet-100, it should require at most the same (if not weaker) data augmentation to improve generalization. Regarding weight decay, similarly, for a smaller dataset, we may require a larger weight decay to avoid overfitting, while a smaller weight decay may be desirable for a larger dataset to avoid underfitting. However, as we will show in the next paragraph, using a larger weight decay brings improvements.

Validating our proposed recipe step-by-step (Table IVa). We find that all three strategies, i.e., using ε warm-up, weak data augmentation, and large weight decay, helps the performance of adversarial training for the XCiT transformer model. Among them, reducing data augmentation yields the highest benefit, while all aspects combined improve the robust accuracy by 12.42%, i.e., by more than 40% relative to the canonical recipe robust accuracy. In comparison, at the time of submission, the leading entry from RobustBench leaderboard for ImageNet-1k [49], by Salman *et al.* [50] achieves 38.14% robust accuracy (3.64% lower than ours) and 68.46% clean accuracy (3.88% lower than ours). Note that Salman *et al.* [50] use WideResNet-50-2, which has 2.6× more parameters and 2.4× more FLOPs than XCiT-S12. Finally, we validate our choice not to use RandomErasing to keep the recipe light: the model trained with it only has 40.60% robust accuracy (1.18% less than without), and comparable clean accuracy (72.42%).

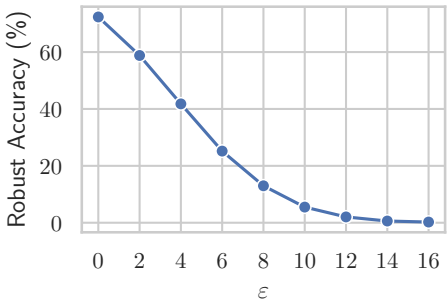


Fig. 3: **AutoAttack is reliable.** We show how the robustness of XCiT-S trained for $\varepsilon = 4/255$ goes down as ε increases. As expected from a reliable attack, the robust accuracy monotonically, but gently decreases with ε.

AutoAttack reliability. To make sure that AutoAttack is a reliable attack for XCiT as much as it is for adversarially trained ResNets (i.e., the model does not suffer from gradient masking or other factors which may impact AutoAttack's performance), we study how the robustness of XCiT-S12 changes when ε increases. As we can see from Fig. 3, the robust accuracy decreases monotonically – but slowly – until it reaches $\sim 0\%$ when $\varepsilon = 16$. This suggests that AutoAttack has no clear issues to find adversarial examples for XCiT. Finally, we note that, among the 5000 validation images used by RobustBench, 2345 are not perturbed successfully by either of APGD-CE and APGD-T [38], and among those 2345, only 1 is perturbed successfully by the FAB-T [51] attack, and among the 2344 remaining images, none is successfully perturbed by the black-box Square attack [52]. The fact that the black-box attack does not manage to decrease the robust accuracy of the white-box ones further suggests that no issues are encountered when computing gradients with white-box attacks. For completeness, we also report PGD-{5, 10, 50, 100} results in Table X in the appendix.

Comparison with Bai *et al.* [2]. In their work, Bai *et al.* [2] observe that training DeiT-S with strong augmentations leads to a collapse of the training procedure, and training with no strong augmentation at all (but with no changes in terms of weight decay) leads to suboptimal performance. On the other hand, they find that increasing the intensity of data augmentation in the first ten epochs stabilizes the training procedure and leads to 35.50% AutoAttack accuracy. However, we observe that, using our implementation, we can train a DeiT-S with heavy augmentation to nontrivial AutoAttack accuracy (32.70%). Our hypothesis for this difference is that we used a 10-epochs learning rate warm-up, instead of a 5-epochs one. Moreover, using our recipe, DeiT-S achieves better robust accuracy than when using the canonical recipe with an increase in data augmentation intensity. Finally, we improve over the best result reported in their work, obtained with a GELU ResNet-50 with both better clean and robust accuracy. The comparison is summarized in Table III.

Our recipe's benefits generalize across architectures (Ta-

TABLE IV: **Success on full scale ImageNet-1k dataset.** We validate our method on the full ImageNet-1k dataset in two steps. In table (a) we step-by-step add each finding from our ablation study (Table II) and shows our bag-of-tricks for ViT's adversarial training generalize to the full ImageNet-1k dataset. The most noticeable improvement comes from using weaker data augmentations, the key finding uncovered in our ablation study. In table (b) we test whether weak data augmentation consistently benefits adversarial training across ViTs or ViTs-inspired architectures. For each network, we consider the training recipe used in the original paper for standard training and compare it with the proposed training recipe. Across all networks, we find that proposed training improves both clean and robust accuracy. We use AutoAttack to measure robust accuracy.

(a) **Step-by-step improvements.** The effect of our training recipe components incrementally when tested on the full ImageNet-1k dataset. Overall, our training recipe improves the robust accuracy by 13.08%, not at the expense of the clean one, which increases by 0.66%.

Feature	Accuracy	
	Clean	AutoAttack
XCiT-S12	71.68	28.70
+ ε warmup (10 epochs)	71.98 (+0.30)	29.36 (+0.66)
+ Tuned data augmentation	71.70 (−0.28)	38.78 (+9.42)
+ Tuned weight decay	**72.34** (+0.64)	**41.78** (+3.00)

(b) **Cross-architecture generalization.** We compare three different architectures adversarially trained on ImageNet-1k with and without heavy data augmentation and small weight decay. For all three architectures, using weak data augmentation brings an advantage, as opposed to standard training, where heavy data augmentation and smaller weight decay bring an advantage. Since for standard training there is no ε involved, the ε *schedule* column applies only to the adversarially trained models. The PoolFormer without data augmentation (marked with *) is trained with 0.05 weight decay, as with 0.5 the training collapses. The symbol † means that the training run collapsed.

Architecture	ε schedule	Heavy data Augmentation	Weight Decay	Adversarial training		Standard training
				Clean accuracy	AutoAttack accuracy	Clean accuracy
XCiT-S12	✗	✓	0.05	71.68	28.70	**80.53**
	✓	✗	0.5	**72.34**	**41.78**	78.96
DeiT-S	✗	✓	0.05	66.30	32.70	**74.61**
	✓	✗	0.5	**66.80**	**37.90**	73.38
ConvNeXt-T	✗	✓	0.05	$0.08^\dagger$	$0.08^\dagger$	**79.87**
	✓	✗	0.5	**71.64**	**44.44**	77.70
PoolFormer-M12	✗	✓	0.05	65.88	34.08	**76.84**
	✓	✗	0.05^*	**66.16**	**34.72**	75.74

ble IVb). In adversarial training, reducing data augmentation strength yields benefits not only for transformer architectures, such as XCiT and PoolFormer [32] but also for the ConvNeXt [34] CNN architecture, as we summarize in Table IVb. With ConvNeXt-T, when using heavy data augmentation as in the original paper, both with, and without, ε warm-up, the training fails. However, when trained using our recipe, i.e., with larger weight decay and weak data augmentation, the model achieves state-of-the-art results. This phenomenon is likely due to the fact that adversarial training requires much higher capacity networks than standard training [4], as it solves the learning objective on all samples within the perturbation budget. While solving it on heavily augmented instances, the network sacrifices network capacity but doesn't yield an equivalent benefit in generalization. On the other hand, from the PoolFormer family, we train PoolFormer-M12, which is both smaller and has fewer FLOPs than ResNet-50, and we observe that, if we use the weight decay from our recipe (0.5) it is extremely unstable. Hence, we use our training recipe with the canonical weight decay (0.05). If we train the same model with full data augmentation from the canonical recipe (the same as the DeiT and XCiT one), we obtain a model with both worse AutoAttack and clean accuracy. Our recipe then brings an improvement also for this architecture, albeit by a smaller margin than ConvNeXt-T.

Our recipe's benefits generalize to the ℓ_2 threat model (Table V). Even though the ℓ_∞ threat model is the scenario that has been studied the most so far for ImageNet [2, 24, 53, 54], it is also important to see whether our recipe generalizes to other threat models, as they may bring additional benefits, such as better standard fine-tuning performance [50]. For this reason, we train XCiT-S12 with both the canonical and our recipe to be robust against ℓ_2-bounded perturbation with $\varepsilon = 3.0$. While the model trained with our recipe has slightly lower clean accuracy, the robust accuracy is better by one-third (i.e., 29.38% vs. 39.94%)., bringing a significant improvement. This suggests that our recipe also generalizes to the ℓ_2 threat model for XCiT-S12. For completeness, we also train a GELU ResNet-50 using the recipe of Bai *et al.* [2] and report the results of the ReLU ResNet-50 shared by Salman *et al.* [50]. Our XCiT-S12 performs better than both, by a substantial margin.

TABLE V: **Performance in the ℓ_2 threat model.** We train XCiT-S12 to be robust against ℓ_2-bounded perturbation with $\varepsilon = 3.0$ employing the canonical recipe and compare it with our recipe. We show that despite the XCiT-S12 trained with our recipe having slightly lower clean accuracy, it has significantly larger robust accuracy. We also show the accuracies of both a GELU ResNet-50 trained according to the set-up of Bai *et al.* [2] and the ReLU ResNet-50 shared by Salman *et al.* [50]. Both have lower performance than the XCiT-S12.

Model	Recipe	Accuracy	
		Clean	AutoAttack
ReLU ResNet-50 [50]	*Canonical*	62.86	34.84
GELU ResNet-50	*Canonical*	66.14	35.60
XCiT-S12	*Canonical*	71.24	29.38
	Ours	70.78	39.94

TABLE VI: **Scaling to larger models.** We test our recipe on larger variants XCiT and compare it to robust ResNets. The XCiT variants outperform ResNets by a wide margin, and achieve *top* rank on the RobustBench [24] benchmark. Finally, we use our training recipe for ConvNeXT [34], PoolFormer [32], and DeiT [30]. We use baseline results from Bai *et al.* [2] and Salman *et al.* [50].

Architecture	Parameters	GFLOPs	Accuracy	
			Clean	AutoAttack
GELU ResNet-50 [2]	25M	4.11	67.38	35.51
WideResNet-50-2 [50]	68M	11.47	68.46	38.14
XCiT-S12	26M	4.82	72.34	41.78
XCiT-M12	46M	8.54	74.04	45.24
XCiT-L12	104M	18.97	73.76	47.60
PoolFormer-M12	22M	3.22	66.16	34.72
DeiT-S	22M	4.61	66.80	37.90
ConvNeXt-T	29M	4.50	71.64	44.44

Validating success on large-scale models (Table VI). Given that our training recipe successfully generalizes across network architectures, we use it to train larger-scale models. For this experiment, we train on a 64-core TPUv4 pod, while scaling batch size and learning rate accordingly. The total training time for XCiT-S12 is 19h30m, for XCiT-M12 it is 33h, and for XCiT-L12 it is 39h. We provide our detailed setup in Appendix D, and we show the progress of the validation FGSM accuracy in Fig. 4. When increasing the scale of the model from S12 to M12, we observe consistent improvement in robust accuracy (from 41.78% to 45.24%). Regarding XCiT-L12, instead, we first observe that, when trained with 1-step FGSM, it achieves sub-par robust accuracy (43.78%). Investigating more, we note that XCiT-L12 has better PGD-10 accuracy than XCiT-M12 (52.22% vs. 51.50%) on the RobustBench ImageNet-1k subset. However, when it comes to APGD-CE (the first of the AutoAttack ensemble), XCiT-L12 is outperformed by XCiT-M12 (46.14% vs. 47.58%). For this reason, we train

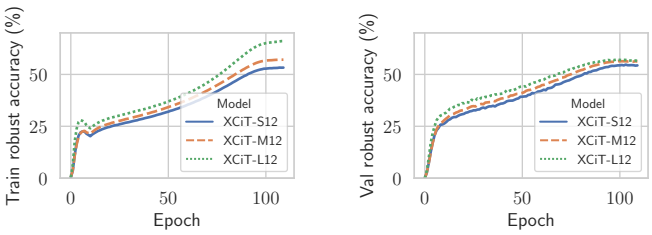


Fig. 4: **Learning curves for the XCiT models.** We show the progress of the training *(Left)* and validation *(Right)* FGSM accuracy for XCiT-{S,M,L}12 trained with our recipe.

XCiT-L12 using 2-step FGSM, which results in a model with better robust accuracy than XCiT-M12 (47.60% vs. 45.24%). Finally, to understand the accuracy-robustness trade-off, we also standardly train these networks for 100 epochs. XCiT-S12, M12, and L12 achieve 80.36%, 81.71%, and 82.65% clean accuracy, respectively. This suggests that adversarial training in ViT sacrifices $\sim 10\%$ clean accuracy to achieve robustness.

Comparison with the EasyRobust library models. Mao *et al.* [53] recently released EasyRobust, a PyTorch library to perform adversarial training. Concurrently to our work, they released several models on GitHub (albeit without a paper that gives information about how they trained the models). These models have been trained with a variety of recipes[4]. In particular, ViT-S has been trained with the *canonical* recipe, while the other models in Table VI have been trained with the same data augmentation we use in our recipe, plus the *Lightning noise* data augmentation, used in the `robustness` library [55]. All the models have been trained for 300 epochs. As they have been trained for $3\times$ the number of epochs as ours, it is not straightforward to draw a comparison. In any case, we note that their top-performing models (when taking into account FLOPs and parameters), i.e., Swin-S (73.41% clean and 46.76% AutoAttack accuracy) and Swin-B (75.05% clean and 47.42% AutoAttack accuracy), have been trained without strong data augmentations and show very promising results. We leave for future research on whether integrating the other elements of our recipe (i.e., ε warm-up and larger weight decay) can further improve the performance of these models.

B. Beyond pre-training: Success of our approach with transfer learning

Setup. We consider four datasets, namely CIFAR-10 and CIFAR-100 [56], Caltech-101 [57], and Oxford flowers [58]. These datasets cover a diverse range in terms of the number of images and image resolution. For all datasets, we use $\varepsilon = 8/255$ in both pre-training and finetuning. Our baselines will be XCiT-S12 and ResNet-50 pre-trained with the canonical recipe, which we compare their success with networks pre-trained with our proposed recipe. Additional details about pre-training, as well as the performance of the resulting models, can be

[4]We discussed the parameters with the authors via e-mail.

found in Appendix D. When finetuning, we use the identical recipe as pre-training for the high-resolution dataset, and a slightly different one, employing TRADES-10 [10], for the low-resolution ones. We provide extensive details on our finetuning procedure in Appendix D.

Success in finetuning on high resolution datasets. We finetune our pre-trained networks on both Caltech-101 and Oxford Flowers for 20 epochs. From Table VIIa we can see that: 1) we can easily fine-tune on both datasets out-of-the-box. 2) Our XCiT achieves the best results, by a significant margin, for both the standardly and adversarially trained models. 3) We can observe that, for XCiT-S12, there is a very good robustness-accuracy trade-off, as the robust model trained on Caltech-101 has a small drop of 2.77% w.r.t. the standardly fine-tuned model (vs. a 5.59% drop for ResNet-50), and the robust model trained on Oxford Flowers has a drop in terms of clean accuracy of 8.79% (vs. an 11.77% drop for ResNet-50). 4) Despite the larger clean accuracy, XCiT-S12 is more robust, having a robust accuracy better by 27.25% on Caltech-101, and 15.16% better on Oxford Flowers. 5) The models pre-trained with our recipe lead to better results across the board when compared to the models pre-trained with the canonical recipe.

Adapting to small resolution images. As we pre-train on the ImageNet-1k dataset at 224×224 resolution, we first need to modify the network architecture to adapt to small resolutions, i.e., 32×32 for CIFAR-10 and CIFAR-100. Previous work [23] achieves this by down-sampling the weights of the convolutional layer that is used by ViT to embed each patch. For XCiT, we adapt the patch-processing module, which converts patches to 1-D vectors, by changing the stride of subsequent convolutions from two to one. We finetune the networks for 20 epochs and report our results in Tables VIIb and VIId. We provide additional details about the models' adaptation and fine-tuning hyper-parameters in Appendix D. To highlight the necessity of pre-training, we also train from scratch on CIFAR-10 using the same setup without pre-training for 300 epochs. We also do a training run using additional synthetic data from Sehwag *et al.* [15] to compensate for the smaller size of the dataset. Moreover, we compare our models to the WideResNet-28-10 by Hendrycks *et al.* [59], who fine-tune a model pre-trained on a sub-sampled version of ImageNet-1k, and to the fine-tuned ResNet-50 pre-trained by Salman *et al.* [50]. Our models perform better than both baselines. In particular, we suspect the performance of the ResNet-50 to be sub-optimal because of the mismatch in resolution, while this factor may not affect XCiT as much. As a matter of fact, XCiT is shown to be more resilient to changes in image resolution [28], while Salman *et al.* [50], when using their pre-trained model for standardly fine-tuning, up-sample both CIFAR datasets. Finally, we note that our XCiT-L model fine-tuned on CIFAR-100, at the time of the submission, would rank second in the corresponding RobustBench leaderboard [24], with better performance than more compute-intensive architectures.

V. Understanding the success of Vision Transformers in adversarial training

So far, we have shown that changing architecture can improve adversarial robustness by a large margin when using an appropriate training recipe. Now, we first investigate a potential reason for this: we hypothesize that, for the top-performing models, few-step attacks are more effective than a conventional ResNet or an XCiT trained with the canonical recipe. This *attack effectiveness* makes the models more robust at test time, as the models have seen stronger attacks during training. After that, we further explore one consequence of robust models: we propose a way to *quantify* the semantic nature of adversarial perturbations. We show that the perturbations targeting a robust XCiT-S have more semantic features than those targeting a robust GELU ResNet-50, reflecting the fact that the robust XCiT-S has larger robust accuracy than the robust ResNet-50.

A. Attack effectiveness influences adversarial training

Adversarial training (Eq. (1)) solves a $\min$-$\max$ optimization where the inner maximization aims to generate strong adversarial examples, while the outer minimization optimizes the network parameters to correctly classify these adversarial examples. The choice of the network architecture simultaneously impacts the optimization success in both $\min$ and $\max$ problems. Better network architectures can certainly achieve better solutions for outer $\min$ problems (e.g., scaling in neural network size leads to better performance). However, their success in adversarial training can simultaneously stem from the ability to achieve a better solution for the inner $\max$ problem, i.e., generate stronger adversarial examples. It is common to use only a few gradient steps [4], even one in some cases [54], to reduce the computational burden of adversarial training. Thus the key question is not whether it's *easier* to generate strong adversarial examples for ViTs but whether it's *easier* under few-steps gradient attacks. Our approach in this direction is not without precedent, as previous work on CNNs observed that better robustness with improved architectural components is because they may make it easier to generate effective adversarial examples in a few steps [1].

Attacking XCiT with gradient-based attacks is as tractable as attacking ResNet. While it is empirically well known that it is moderately tractable to optimize adversarial examples with gradient-based methods on adversarially trained ResNets [4], to the best of our knowledge this has not been studied in the case of adversarially trained ViTs. For this reason, we compute the loss given by separate PGD attacks ran with a different number of steps (1, 5, 10, 50, 100, 200, 500), scaling the attack step size accordingly: the maximum loss is reached by attacks that use at least 100 steps. We run this experiment with twenty different random restarts for each point to see if they all converge to similar maxima: all the runs for each point show extremely similar loss curves, for both the robust XCiT-S12 and the robust ResNet-50 (Fig. 5). Finally, in Fig. 6 we show how the loss changes at every step during twenty separate PGD-500 runs, each starting from a different random

TABLE VII: **Advantages of the pre-training recipe also directly transfer to fine-tuning.** When finetuning results on both high and low-resolution datasets, we find that our proposed recipe achieves better performance. The XCiT-S12 marked with *(c)* is the one pre-trained using the canonical recipe used for standard training, while *(ours)* refers to models pre-trained with our proposed recipe. Both models are adapted by changing the stride of the initial convolution to change the patch size. Note that pre-training and finetuning methods (standard/adversarial) are kept identical for Caltech-101 and Oxford Flowers. The ResNet-50 pre-trained checkpoint is from Salman *et al.* [50], and is adapted to small resolutions by changing the stride of the first convolution layer, and the WideResNet-28-10 in Tables VIIb and VIId are from Hendrycks *et al.* [59].

(a) **Fine-tuning on high-resolution datasets.**

Fine-tuning	Model	Dataset			
		Caltech-101		Oxford Flowers	
		Clean	AutoAttack	Clean	AutoAttack
Standard	ResNet-50	86.97	7.56	86.28	1.96
	XCiT-S12 *(c)*	89.92	0.96	88.13	0.99
	XCiT-S12 *(ours)*	**90.36**	**17.12**	**91.65**	**5.71**
Adversarial	ResNet-50	81.38	34.49	74.51	32.75
	XCiT-S12 *(c)*	86.18	58.84	76.26	42.42
	XCiT-S12 *(ours)*	**87.59**	**61.74**	**82.86**	**47.91**

(b) **CIFAR-10 adversarial fine-tuning.**

Model	Clean Accuracy	AA Accuracy
WideResNet-28-10 [59]	87.11	54.92
ResNet-50	84.80	41.56
XCiT-S12 *(c)*	89.07	54.37
XCiT-S12 *(ours)*	90.06	56.14
XCiT-M12 *(ours)*	91.30	57.27
XCiT-L12 *(ours)*	**91.73**	**57.58**

(c) **Pre-training is necessary for smaller datasets.** Fine-tuning performance when a XCiT-S12 model is 1) trained from scratch 2) trained from scratch with extra synthetic data [15] 3) pre-trained on ImageNet-1k.

Pre-training	Synthetic data	Accuracy	
		Clean Accuracy	AA Accuracy
✗	✗	82.84	39.49
✗	✓	80.01	47.88
✓	✗	**90.06**	**56.14**

(d) **CIFAR-100 adversarial fine-tuning.**

Model	Clean Accuracy	AA Accuracy
WideResNet-28-10 [59]	59.23	28.42
ResNet-50	61.28	22.01
XCiT-S12 *(c)*	65.44	30.97
XCiT-S12 *(ours)*	67.34	32.19
XCiT-M12 *(ours)*	69.21	34.21
XCiT-L12 *(ours)*	**70.76**	**35.08**

start, targeting the same point, with a relatively large step size. This shows that the attack converges after a few steps. We show more results for both experiments on thirty-one more random samples in Appendix I.

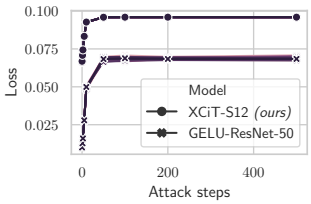


Fig. 5: **Is PGD-200 a good Oracle?** Saturating of the cross-entropy loss in separate runs of PGD attacks with different numbers of steps, perturbing the same input. Plots for additional points are in Appendix I.

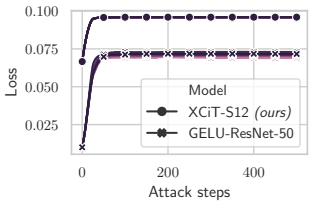


Fig. 6: **Attacks with different initialization converge to very similar losses.** Evolution of the loss for different runs of PGD attacks perturbing the same input, using a large step size. Plots for additional points are in Appendix I.

Effectiveness of k-step attacks. We use the following metric (d_k) to measure the tractability of the inner maximization problem with a k-step gradient attack, which we call *attack effectiveness*:

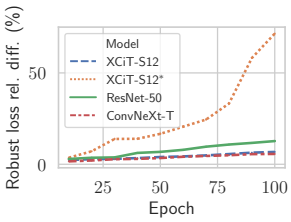
$$d_k = \frac{\mathcal{L}(\mathbf{x} + \boldsymbol{\delta}_k, \mathrm{y}; \boldsymbol{\theta}) - \mathcal{L}(\mathbf{x} + \boldsymbol{\delta}_O, \mathrm{y}; \boldsymbol{\theta})}{\mathcal{L}(\mathbf{x} + \boldsymbol{\delta}_O, \mathrm{y}; \boldsymbol{\theta})} \tag{2}$$

where $\boldsymbol{\delta}_k$ is the perturbation generated with k step attacks while $\boldsymbol{\delta}_O$ is the perturbation generated with an *Oracle*, i.e., a strong attack. As commonly done, we use cross-entropy loss ($\mathcal{L}$). This metric measures the strength of a k-step attack compared to the strongest attack. As validated previously, using a very high number of attack iterations appears to find the local optimum. Thus we use 200 attack steps as an Oracle. We measure the effectiveness of adversarial examples generated with PGD-$\{1, 2, 5, 10\}$ attacks. We use the full ImageNet-1k validation dataset (50,000 images) for this experiment.

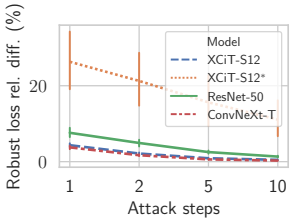
Few step attacks are highly effective for the most successful models. Throughout training progress (epochs 1 to 100), we find that a single-step attack is more effective against our trained XCiT than a ResNet-50 network (Fig. 7a). A similar trend is observed when we ablate across the number of attack steps (Fig. 7b). This does not hold just for transformers: we observe that the modern ConvNeXt model, which is as successful as XCiT, also enjoys high effectiveness of few-steps attacks. Both observations suggest that the ease of optimizing the inner $\max$ problem with few-step attacks does indeed heavily impact the success of a model in adversarial training.

Attack effectiveness in proposed vs canonical training recipe. Since the $\min$ and $\max$ problems are solved alternatively, i.e., network parameters are continuously updated, the ability the generate strong adversarial examples under fixed steps attacks would vary over time. To analyze this phenomenon, we compare the effectiveness of attacks when attacking the XCiT-S12 trained with the canonical training recipe (with just the ε warm-up from our recipe): we observe that when using the canonical recipe, the attacks are extremely ineffective, also compared to ResNet's training. This shows that the way we perform the (outer) $\min$ optimization of adversarial training influences the ease of (inner) $\max$ optimization, and an effective training recipe is crucial for successful training.

Discussion. Intuitively, one could argue that it's easy to generate attacks for poorly trained models, and these models show no robust accuracy. However, in our case, the XCiT-S12 trained with our recipe, not only has nontrivial performance, but the clean accuracy is better than that of the XCiT-S12 trained with the canonical recipe. Nonetheless, few-steps attacks generate more powerful adversarial examples for these models, and doing more steps does not bring a significant advantage. On the other hand, for other models, such as the XCiT-S12 trained with the canonical recipe or the GELU ResNet-50, doing more attack steps brings a larger advantage, suggesting that for these models the inner $\max$ optimization is harder.



(a) Relative difference between the adversarial loss computed with PGD-1 and the one computed with PGD-200, varying over the training epochs.



(b) Relative difference between the adversarial losses computed with PGD attacks with 1, 2, 5, and 10 steps, and the adversarial loss computed with PGD-200, averaged across 10 epochs.

Fig. 7: **Few-step attacks of the more robust models are more effective throughout the training**. We test whether the higher robustness of models relates to the ease of optimization of adversarial loss, i.e., using few-step attacks during adversarial training. We measure the relative difference in the success of a few-step attack, i.e., weak but fast, compared to a strong one. We can observe that the relative difference is smaller for the more robust models. This suggests that these few-step attacks (with often one or two steps) are more effective for XCiT-S and ConvNeXt, hence making the final models more robust. We can also observe a significant difference between the XCiT-S12 trained with our recipe and the one trained with the canonical recipe (marked as "XCiT-S12*"). These results are computed on the ImageNet-1k validation set and the confidence intervals are over three runs.

B. Semantic nature of XCiT's adversarial perturbations

Earlier works have demonstrated that adversarial perturbations for robust CNNs have semantic, interpretable patterns rather than unintelligible noise as in the case of non-robust networks [35]. Given the different nature of the ViTs architectures such as XCiT, compared to CNNs, it is natural to ask whether a similar characteristic also emerges for these architectures. Hence, we explore the perturbations targeting a robust XCiT and compare them with those targeting a non-robust XCiT from `timm` [60], and those targeted to the robust GELU ResNet-50 by Bai *et al.* [2].

Quantifying the semantic nature of perturbations. To quantify how semantic the perturbations are, we propose to classify untargeted adversarial perturbations with high-performing, standardly trained models. We hypothesize that if a perturbation is semantic enough (i.e., tries to change the nature of the input from the point of view of the human eye), then it should be classified with the class it tries to evade, as the perturbations should be focused on the shapes characterizing such class in the input image. We use, as classifiers, ConvNeXt-XL [34], with 87.01% clean accuracy, BeiT-L [61], with 87.48% clean accuracy, and Swin-B [62], with 86.32% clean accuracy on ImageNet-1k, using the implementation and pre-trained weights from the timm library [60]. All the models accept input size 224×224. We generate PGD-100 perturbations for the 5000 images subset of RobustBench for our robust XCiT-S12 and a non-robust XCiT-S12 with pre-trained weights from timm [60], as well as for a robust ResNet-50 from Bai *et al.* [2] and a non-robust ResNet-50 from the timm library (shared by Wightman *et al.* [42]). We scale the perturbations into the $[0, 1]$ range. We show some example images of perturbations in Fig. 8. We can see from Table VIII that, according to this metric, the perturbations generated for both robust models lead to non-trivial accuracies and that the perturbations generated for XCiT-S12 have indeed semantic characteristics, and more so than those generated for the robust ResNet-50. We provide additional feature visualizations in Appendix J.

TABLE VIII: **Semantic characteristics in adversarial perturbations.** For each robust network, we first generate adversarial perturbations ($\varepsilon = \frac{4}{255}$, ℓ_∞, scaled to [0, 1]) using untargeted attack. We provide an example visualization of these perturbations in Figure 8. Measuring the top-5 accuracy using pre-trained ConvNeXt-XL, BeiT-L, and Swin-L models, we observe that robust XCiT-S perturbations indeed have semantic characteristics, even higher than a robust ResNet-50.

Perturbations generator		Classifier		
		ConvNeXt-XL [34]	BeiT-L [61]	Swin-L [62]
Robust	XCiT-S12 *(ours)*	**43.86**	**49.52**	**40.24**
	GELU ResNet-50 [2]	38.40	45.02	36.70
Non-robust	XCiT-S12 [28]	0.84	0.78	0.84
	ResNet-50 [42]	0.82	0.74	0.80

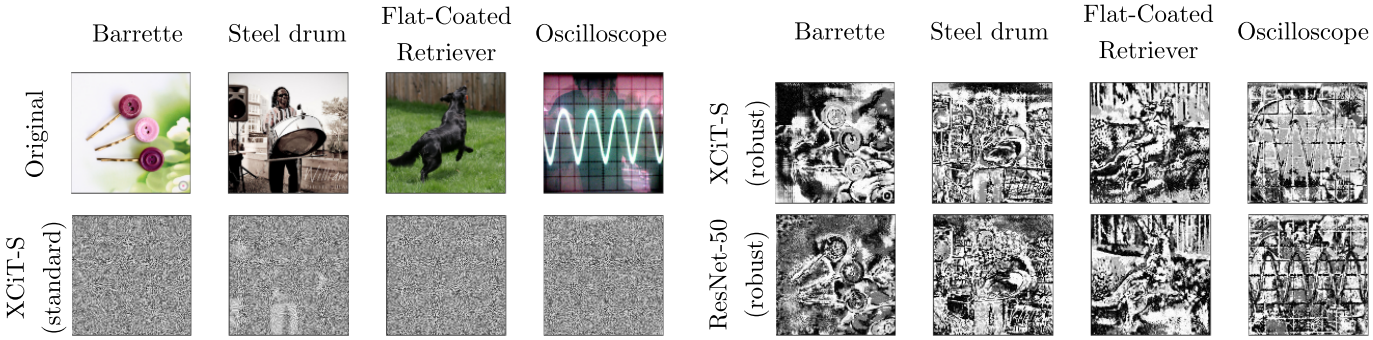


Fig. 8: **Example adversarial perturbations for robust vs. non-robust models.** Comparison between the adversarial perturbations generated for a robust XCiT-S12, a robust GELU ResNet-50 by Bai *et al.* [2] and a regular XCiT-S12 from the timm [60] library. The perturbations, generated with a ℓ_∞ attack (ε=$\frac{4}{255}$), are scaled to the $[0, 1]$ range, and, for the sake of visualization, transformed into black and white images. Similar to robust CNNs [35], as demonstrated for a ResNet-50 here, robust XCiT perturbations also exhibit semantic characteristics in its adversarial perturbations. When quantitatively compared, we find that XCiT adversarial perturbations have even more semantic information than the robust ResNet-50 perturbation.

VI. Related work

Vision Transformers and variants The ViT [18] is an architecture that was adapted for computer vision tasks from the transformer [39], which was first meant for natural language processing. After the introduction of the transformer, several variants have been proposed. Some of these are: DeiT [30], to reduce the need for pre-training on extremely large datasets, CaiT [31], to train deeper ViTs, XCiT [28], to use a more efficient attention-like operation, LeViT [33], a CNN-ViT hybrid to speed-up transformer inference, and the Swin Transformer [62], which is a hierarchical transformer which can perform image segmentation and object detection.

Robustness of Vision Transformers to non-adversarial perturbations. Whether ViTs are more robust than CNNs has been a controversial topic so far, with contrasting results based on the different contexts and settings where the models are tested. On the one hand, many recent works [2, 19, 20] agree on the fact that ViTs are more robust than CNNs when it comes to out-of-distribution samples. Although, more recently, Pinto *et al.* [63] find that ViTs suffer from simplicity bias similarly to CNNs [64], and that, when compared to modern CNNs such as ConvNeXt, there is no clear winner.

Adversarial robustness of Vision Transformers At the same time as they assess ViTs' robustness to natural perturbations, many works also study the robustness of non-adversarially trained ViTs to adversarial examples. In particular, in the case of attacks with $\varepsilon \leq 0.01$, ViTs show better robustness than CNNs [20, 21, 65]. However, when running stronger attacks, such as APGD or AutoAttack, with $\varepsilon = 4/255$, both ResNets and ViTs have 0% robust accuracy [2, 23, 66]. Finally, Mao *et al.* [67] propose the *Robust ViT* (RVT), a ViT with specific architectural innovations which show additional robustness to FGSM and PGD-5 attacks. However, they do not test the robustness of RVT to stronger attacks such as AutoAttack.

Adversarially trained Vision Transformers. Shao *et al.* [23] adversarially fine-tuned non-robust ViTs, pre-trained on ImageNet-1k. Since they use a non-robust pre-trained network, they achieve suboptimal performance [59]. Bai *et al.* [2] comes closest to our work: they noticed poor performance of DeiT-S transformers with data augmentation in adversarial training. We show that similar challenges persist with other ViT architectures, such as XCiTs. While Bai *et al.* [2] advocate for a progressively increasing augmentation budget, we show that avoiding strong data augmentation entirely can achieve state-of-the-art performance. Even further, we perform an in-depth analysis of adversarially robust ViTs in both pre-training and finetuning. Wu *et al.* [22] propose a technique to make adversarial training of transformers more efficient. This mechanism reduces the training time while increasing the robust accuracy over 1-step FGSM. However, since they use the canonical training recipe from DeiT [30], the robust accuracy of their models could be sub-optimal. Finally, concurrently with this work, Mao *et al.* [53] published the checkpoints of several ViTs who are adversarially trained and show promising performance. However, the training recipe for their models is not public and there is no paper discussing their results.

Robustness through architectural development. Xie *et al.* [1] observe that, in ResNets and other CNN variants, the usage of smooth activation functions, such as SiLU [68] and GELU [3] increases the robustness of adversarially trained models: their hypothesis is that smoother activation functions make the inner maximization problem in Eq. (1) easier to solve. This behavior is also observed by Bai *et al.* [2]: ResNet-50 is more robust when employing GELU. Huang *et al.* [17] observe that increasing the capacity of a model helps with robustness, but there is a trade-off given by the fact that additional capacity makes the model less smooth and less robust. However, reducing the capacity of the last stage can improve this trade-off. Similar observations are done in concurrent work by Wu *et al.* [69] who propose a technique to efficiently find a suitable TRADES' λ when training wide models.

Improving the robustness-accuracy-efficiency trade-off. Apart from the work by Wu *et al.* [22] mentioned above, other previous work addresses the problem of efficiency of robustness in different ways. One line of work focuses on making adversarial training more efficient: Shafahi *et al.* [70] proposes to re-use adversarial examples from previous epochs, while Wong *et al.* [54] find an effective way to perform adversarial training with just 1-step FGSM. On the other hand, Sehwag *et al.* [71] and Kundu *et al.* [72] work on compressing the model's size by pruning in ways that are fully compatible with adversarial training. Other works show that both robustness and accuracy are improved when we use either extra unlabeled data [73] or synthetic data [14, 15].

VII. Discussion and Conclusion

Architectures and custom recipes. This work shows that, by shifting architecture, we can significantly improve the robustness of image classification models, by keeping a good accuracy-robustness-efficiency tradeoff. We do so by identifying an architecture that has a good fit for adversarial training: the Cross-Covariance Image Transformer (XCiT). We also show that to achieve optimal results, it is important to find a tailored training recipe, which may differ from the canonical training recipe for standard training. Using this custom recipe, we achieve good results which are better than the current state-of-the-art, both in terms of clean and robust accuracy. On the other hand, we have also tested this training recipe on ConvNeXt, a recently proposed modern convolutional architecture, showing that, with our training recipe, it can reach state-of-the-art performance. We leave to future research whether the performance can be further boosted with a custom-tailored recipe for ConvNeXt and the architectural innovations to come (both for training from scratch and fine-tuning).

Fine-tuning. We further successfully show that ViTs can be efficiently robustly fine-tuned, for larger perturbation sizes, to very high accuracy on smaller datasets. As a matter of fact, our tailored training recipe also works for larger perturbations with minimal changes. This enables efficiently fine-tuning these models to other datasets and doing adversarial training on smaller high-resolution datasets. Moreover, given the trends shown in standard training of ViTs and previous work about adversarial training, we believe that XCiT could further benefit from being trained on larger datasets such as ImageNet-21k and then fine-tuned on downstream datasets. We suggest that researchers should consider this option when doing adversarial training for ViT-like models, given that, as we have shown, a model can be fine-tuned efficiently in a few epochs.

Analyses. We show a potential explanation of the improved robustness of XCiT and ConvNeXt compared to ResNet and XCiT trained with the canonical recipe: for former models, the 1-step attack is more effective throughout the whole training procedure. Finally, we analyze the gradients of our robust XCiT and compare the visualizations to a state-of-the-art robust ResNet: we quantify that the perturbations found for XCiT are more semantic than those of ResNet, suggesting that the robust XCiT's perturbations are more aligned with human perception. We believe that further insightful analyses can be carried on, given the different nature of ViT-like models. For this reason, we release the checkpoints of our models trained for different epsilons. We believe that this enables researchers to do further analyses that will improve our understanding of why such a simple recipe is particularly suitable for adversarial training.

References

[1] C. Xie, M. Tan, B. Gong, A. Yuille, and Q. V. Le, "Smooth adversarial training," *arXiv preprint arXiv:2006.14536*, 2020.

[2] Y. Bai, J. Mei, A. L. Yuille, and C. Xie, "Are transformers more robust than cnns?" *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[3] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *arXiv preprint arXiv:1606.08415*, 2016.

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

[5] B. Biggio *et al.*, "Evasion attacks against machine learning at test time," in *Joint European conference on machine learning and knowledge discovery in databases*, Springer, 2013, pp. 387–402.

[6] C. Szegedy *et al.*, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

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

[8] S. Lee, H. Lee, and S. Yoon, "Adversarial vertex mixup: Toward better adversarially robust generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 272–281.

[9] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," *Advances in neural information processing systems*, vol. 31, 2018.

[10] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International conference on machine learning*, PMLR, 2019, pp. 7472–7482.

[11] Y. Wang, D. Zou, J. Yi, J. Bailey, X. Ma, and Q. Gu, "Improving adversarial robustness requires revisiting misclassified examples," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rklOg6EFwS.

[12] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Advances in Neural Information Processing Systems*, vol. 33, pp. 2958–2969, 2020.

[13] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. A. Mann, "Data augmentation can improve robustness," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[14] S. Gowal, S.-A. Rebuffi, O. Wiles, F. Stimberg, D. A. Calian, and T. A. Mann, "Improving robustness using generated data," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[15] V. Sehwag *et al.*, "Robust learning meets generative models: Can proxy distributions improve adversarial robustness?" *arXiv preprint arXiv:2104.09425*, 2021.

[16] S. Dai, S. Mahloujifar, and P. Mittal, "Parameterizing activation functions for adversarial robustness," in *2022 IEEE Security and Privacy Workshops (SPW)*, IEEE, 2022, pp. 80–87.

[17] H. Huang, Y. Wang, S. Erfani, Q. Gu, J. Bailey, and X. Ma, "Exploring architectural ingredients of adversarially robust deep neural networks," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[18] A. Dosovitskiy *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *ArXiv*, vol. abs/2010.11929, 2021.

[19] S. Paul and P.-Y. Chen, "Vision transformers are robust learners," *arXiv preprint arXiv:2105.07581*, 2021.

[20] S. Bhojanapalli, A. Chakrabarti, D. Glasner, D. Li, T. Unterthiner, and A. Veit, "Understanding robustness of transformers for image classification," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10 231–10 241.

[21] A. Aldahdooh, W. Hamidouche, and O. Deforges, "Reveal of vision transformers robustness against adversarial attacks," *arXiv preprint arXiv:2106.03734*, 2021.

[22] B. Wu, J. Gu, Z. Li, D. Cai, X. He, and W. Liu, "Towards efficient adversarial training on vision transformers," *arXiv preprint arXiv:2207.10498*, 2022.

[23] R. Shao, Z. Shi, J. Yi, P.-Y. Chen, and C.-J. Hsieh, "On the adversarial robustness of vision transformers," *arXiv preprint arXiv:2103.15670*, 2021.

[24] F. Croce *et al.*, "Robustbench: A standardized adversarial robustness benchmark," *arXiv preprint arXiv:2010.09670*, 2020.

[25] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah, "Transformers in vision: A survey," *ACM Computing Surveys (CSUR)*, 2021.

[26] *Image classification on imagenet (papers with code)*, [Accessed Sep 1, 2022], 2022. [Online]. Available: https://paperswithcode.com/sota/image-classification-on-imagenet.

[27] A. P. Steiner, A. Kolesnikov, X. Zhai, R. Wightman, J. Uszkoreit, and L. Beyer, "How to train your vit? data, augmentation, and regularization in vision transformers," *Transactions of Machine Learning Research*, 2022. [Online]. Available: https://openreview.net/forum?id=4nPswr1KcP.

[28] A. El-Nouby *et al.*, "Xcit: Cross-covariance image transformers," *ArXiv*, vol. abs/2106.09681, 2021.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition. arxiv 2015," *arXiv preprint arXiv:1512.03385*, 2015.

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou, "Training data-efficient image transformers & distillation through attention," in *International Conference on Machine Learning*, PMLR, 2021, pp. 10 347–10 357.

[31] H. Touvron, M. Cord, A. Sablayrolles, G. Synnaeve, and H. J'egou, "Going deeper with image transformers," *ArXiv*, vol. abs/2103.17239, 2021.

[32] W. Yu *et al.*, "Metaformer is actually what you need for vision," *arXiv preprint arXiv:2111.11418*, 2021.

[33] B. Graham *et al.*, "Levit: A vision transformer in convnet's clothing for faster inference," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 259–12 269.

[34] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[35] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, B. Tran, and A. Madry, "Adversarial robustness as a prior for learned representations," *arXiv preprint arXiv:1906.00945*, 2019.

[36] R. Rade and S.-M. Moosavi-Dezfooli, "Helper-based adversarial training: Reducing excessive margin to achieve a better accuracy vs. robustness trade-off," in *ICML 2021 Workshop on Adversarial Machine Learning*, 2021. [Online]. Available: https://openreview.net/forum?id=BuD2LmNaU3a.

[37] L. Rice, E. Wong, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, PMLR, 2020, pp. 8093–8104.

[38] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," *ArXiv*, vol. abs/2003.01690, 2020.

[39] A. Vaswani *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[40] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[41] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2009.

[42] R. Wightman, H. Touvron, and H. Jégou, "Resnet strikes back: An improved training procedure in timm," *arXiv preprint arXiv:2110.00476*, 2021.

[43] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[44] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong

classifiers with localizable features," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[45] E. D. Cubuk, B. Zoph, J. Shlens, and Q. V. Le, "Randaugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[46] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[47] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, "Random erasing data augmentation," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 13 001–13 008.

[48] T. Pang, X. Yang, Y. Dong, H. Su, and J. Zhu, "Bag of tricks for adversarial training," *arXiv preprint arXiv:2010.00467*, 2020.

[49] F. Croce *et al.*, *RobustBench website*, [Accessed Sep 1, 2022], 2021. [Online]. Available: https://robustbench.github.io.

[50] H. Salman, A. Ilyas, L. Engstrom, A. Kapoor, and A. Madry, "Do adversarially robust imagenet models transfer better?" *Advances in Neural Information Processing Systems*, vol. 33, pp. 3533–3545, 2020.

[51] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *Proceedings of the 37th International Conference on Machine Learning*, H. D. III and A. Singh, Eds., ser. Proceedings of Machine Learning Research, vol. 119, PMLR, Jun. 2020, pp. 2196–2205. [Online]. Available: https://proceedings.mlr.press/v119/croce20a.html.

[52] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *European Conference on Computer Vision*, Springer, 2020, pp. 484–501.

[53] X. Mao *et al.*, *EasyRobust: A large-scale robust training toolkit*, https://github.com/alibaba/easyrobust/tree/4c0ec0b0c908004b5c65f718de43a530a0856366, [Online; accessed 24-August-2022], 2022.

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

[55] L. Engstrom, A. Ilyas, H. Salman, S. Santurkar, and D. Tsipras, *Robustness (python library)*, 2019. [Online]. Available: https://github.com/MadryLab/robustness.

[56] A. Krizhevsky, "Learning multiple layers of features from tiny images," University, Tech. Rep., 2009.

[57] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Pattern Recognition Workshop*, 2004.

[58] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, Dec. 2008.

[59] D. Hendrycks, K. Lee, and M. Mazeika, "Using pre-training can improve model robustness and uncertainty," in *Proceedings of the 36th International Conference on Machine Learning*, K. Chaudhuri and R. Salakhutdinov, Eds., ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 2712–2721. [Online]. Available: https://proceedings.mlr.press/v97/hendrycks19a.html.

[60] R. Wightman, *Pytorch image models*, https://github.com/rwightman/pytorch-image-models, 2019. DOI: 10.5281/zenodo.4414861.

[61] H. Bao, L. Dong, S. Piao, and F. Wei, "BEit: BERT pre-training of image transformers," in *International Conference on Learning Representations*, 2022. [Online]. Available: https://openreview.net/forum?id=p-BhZSz59o4.

[62] Z. Liu *et al.*, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2021, pp. 10 012–10 022.

[63] F. Pinto, P. H. Torr, and P. K. Dokania, "An impartial take to the cnn vs transformer robustness contest," *arXiv preprint arXiv:2207.11347*, 2022.

[64] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 9573–9585. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/6cfe0e6127fa25df2a0ef2ae1067d915-Paper.pdf.

[65] P. Benz, S. Ham, C. Zhang, A. Karjauv, and I. S. Kweon, "Adversarial robustness comparison of vision transformer and mlp-mixer to cnns," *arXiv preprint arXiv:2110.02797*, 2021.

[66] K. Mahmood, R. Mahmood, and M. Van Dijk, "On the robustness of vision transformers to adversarial examples," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 7838–7847.

[67] X. Mao *et al.*, "Towards robust vision transformer," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 042–12 051.

[68] S. Elfwing, E. Uchibe, and K. Doya, "Sigmoid-weighted linear units for neural network function approximation in reinforcement learning," *Neural Networks*, vol. 107, pp. 3–11, 2018.

[69] B. Wu, J. Chen, D. Cai, X. He, and Q. Gu, "Do wider neural networks really help adversarial robustness?" *Advances in Neural Information Processing Systems*, vol. 34, pp. 7054–7067, 2021.

[70] A. Shafahi *et al.*, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[71] V. Sehwag, S. Wang, P. Mittal, and S. Jana, "Hydra: Pruning adversarially robust neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 19 655–19 666, 2020.

[72] S. Kundu, M. Nazemi, P. A. Beerel, and M. Pedram, "A tunable robust pruning framework through dynamic network rewiring of dnns," *arXiv preprint arXiv:2011.03083*, 2020.

[73] Y. Carmon, A. Raghunathan, L. Schmidt, J. C. Duchi, and P. S. Liang, "Unlabeled data improves adversarial robustness," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[74] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 843–852.

[75] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[76] E. Hoffer, T. Ben-Nun, I. Hubara, N. Giladi, T. Hoefler, and D. Soudry, "Augment your batch: Improving generalization through instance repetition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8129–8138.

[77] C. Herrmann *et al.*, "Pyramid adversarial training improves vit performance," *arXiv preprint arXiv:2111.15121*, 2021.

APPENDIX

A. The vision transformer architecture

We now cover the basic building blocks of the (vision) transformer architecture: attention and multi-head attention, the MLP block, positional encoding, and tokens embedding.

1) Attention: The attention function maps a query vector and a set of key-value vector pairs to an output vector. In particular, the form of attention by Vaswani *et al.* [39], called *Scaled Dot-Product Attention*, can be formally expressed as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{3}$$

where $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ represent respectively the set of queries, keys, and values grouped in matrices, and d_k represents the dimension of the queries and the keys, while the values have dimension d_v. The product is scaled by $\frac{1}{\sqrt{d_k}}$ to compensate for the fact that the dot products can reach large values in magnitude when d_k is large. Large values would saturate softmax and make its gradients very small. In practice, in the context of NLP, attention is computed among different words (or parts of words). Given a set of words (e.g., a sentence), it measures how much a word is dependent on another word in the same set. For instance, in the sentence "This is a paper about Vision Transformers", "this" will attend to "is", which will, in turn, attend to "paper". The main advantage of attention, when compared to convolutions, is the ability to capture long-distance dependencies between tokens, which is something convolutions fail to do because of the local nature of the convolution operator. A potential advantage of using attention for computer vision tasks is that it can measure how much a portion of an image attends to another one, enabling the possibility of working more at a global level than a local one, as convolutions do. For instance, in an image of a cat, the tail, or the paws, will attend to the cat's head, and vice-versa.

2) Multi-Head Attention: Before passing $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ to the Attention function, Vaswani *et al.* [39] linearly project, using learnable matrices, the inputs into vectors with dimension d_k, d_k, and d_v respectively. Moreover, instead of doing it just once, they do it h times, and each projection is passed to the Attention function simultaneously, creating the so-called Multi-Head Attention. After the parallel processing, the resulting matrices are concatenated and linearly projected, using, again, a learnable matrix. Parallel processing enables the model to efficiently process information from different representations of the inputs. Formally,

$$\begin{aligned}\text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}^O, \\ \text{where head}_i &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V),\end{aligned} \tag{4}$$

where $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, and $\boldsymbol{W}^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are the learnable matrices used to linearly project the inputs and the result of the Attention operation. Finally, we call Self-Attention the special case where $\boldsymbol{K} = \boldsymbol{V}$, and –analogously– Multi-Head Self-Attention (MSA) a Multi-Head Attention in the case where $\boldsymbol{K} = \boldsymbol{V}$.

3) The MLP block and the overall Transformer block: After computing self-attention for the inputs, the result is passed to a fully connected Multi-Layer Perceptron (MLP) block with one hidden layer. Formally, given an input x, and learnable weights and biases $\boldsymbol{W}_1$, $\boldsymbol{b}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_2$, and an activation function ρ, the MLP block can be expressed as

$$\text{MLP}(\boldsymbol{x}) = \rho(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2. \tag{5}$$

The vision transformer uses the Gaussian Error Linear Unit (GELU) [3] activation and apply layer normalization to the input of each MSA and MLP block.

To summarize, given an input $\boldsymbol{x}_l$ to the l-th Vision Transformer block, a multi-head self-attention block MSA, an MLP block, and a layer normalization block LN, the overall Transformer block can be expressed as:

$$\begin{aligned}\boldsymbol{x}_l' &= \boldsymbol{x}_l + \text{MSA}(\text{LN}(\boldsymbol{x}_l)) \\ \boldsymbol{x}_{l+1} &= \boldsymbol{x}_l' + \text{MLP}_{\text{GELU}}(\text{LN}(\boldsymbol{x}_l')).\end{aligned} \tag{6}$$

4) Positional encoding: The reader can observe that attention, per se, considers the input as a set, and not as a sequence. Hence, all information about the position of the inputs is completely lost. For this reason, Vaswani *et al.* [39] add the so called Positional Encodings to the input embeddings before feeding them to the encoder and the decoder. The positional encodings have the same size as the embeddings of the input tokens, i.e., d_{model}. In the case of the vision transformer, the positional embeddings are learned parameters.

5) Input tokenization and positional encoding: Considering each pixel as a token and computing attention between every pixel would be computationally unfeasible, as the attention operation has $\mathcal{O}(n^2)$ complexity for both memory and runtime. For this reason, Dosovitskiy *et al.* [18] split the image into non-overlapping input patches. The patches are embedded into tokens by reducing, by the size of the patches, the overall number of inputs to the attention operation. The patches are embedded by linearly projecting them to vectors of dimension $\mathbb{R}^d_{\text{model}}$. Given an image of size (H, W) and patches of size (P, P), the resulting number of patches is $N = HW/P^2$.

6) Class token: After the input tokens are generated, a vector — called `[class]` token — is prepended to the sequence of tokens and processed along with the other tokens. The initial state of this vector is a learnable parameter of the model. The class token is meant to attend to the most relevant parts of an image, e.g., in the case of an image with a cat, the `[class]` token will attend to the patches, including the cat's head, its tail, and its whiskers. Finally, at the end of the last block, there is the so-called "MLP Head": an MLP which takes as input the `[class]` token resulting from the last attention block, and maps it to the class predicted for the input.

7) Performance and variations of Vision Transformers: ViTs achieve state-of-the-art performance on several datasets, such as ImageNet-1k [41], CIFAR-10, and CIFAR-100 [56]. In particular, their maximum potential is reached when they are pre-trained on larger datasets, such as ImageNet-21k and JFT-300M [74]. In this way, they can learn representations that are more generalizable and do not overfit when they are trained on smaller datasets such as CIFAR-10 and CIFAR-100. To reduce the need for pre-training on larger datasets, concurrent work Steiner *et al.* [27] and Touvron *et al.* [30] shows that a tuned training recipe, strong regularization, and data augmentations, such as CutMix [44], RandAugment [45], MixUp [46], and RandomErasing [47], lead to a ten-fold decrease in the need of data to achieve the same performance. In particular, Touvron *et al.* [30] call the model trained with this training recipe *DeiT*. On the architectural point of view, instead, it has been proposed to specialize layers for patches processing and classification separately, with the aim of more effectively train deeper ViTs [31], and to use different operations than multi-head self-attention, such as cross-covariance attention [28] or average pooling [32].

B. Transformer variations

We now give an overview of two transformer architectures we employ in our experiments, i.e., CaiT [31] and XCiT [28].

1) Class Attention in Transformers (CaiT): In order to train deeper ViTs, Touvron *et al.* [31] introduce two innovations: *LayerScale* and *Class Attention*.

a) LayerScale: LayerScale consists of two learnable diagonal matrices: one is multiplied to the result of the attention operation, and the other is multiplied to the result of the MLP block. Formally, given the two LayerScale diagonal matrices $\text{diag}(\lambda_{l,1}, \dots \lambda_{l,d})$ and $\text{diag}(\lambda'_{l,1}, \dots \lambda'_{l,d})$, the transformer block function becomes

$$\begin{aligned} \boldsymbol{x}'_l &= \boldsymbol{x}_l + \text{diag}(\lambda_{l,1}, \dots \lambda_{l,d})\, \text{MSA}(\text{LN}(\boldsymbol{x}_l)) \\ \boldsymbol{x}_{l+1} &= \boldsymbol{x}'_l + \text{diag}(\lambda'_{l,1}, \dots \lambda'_{l,d})\, \text{MLP}(\text{LN}(\boldsymbol{x}'_l)). \end{aligned} \tag{7}$$

In the case of deeper architectures (i.e., with a total of 24 transformer blocks), these diagonal matrices are initialized to a value ε. This value is equal to 0.1 for architectures with up to depth 18, 10^{-5} for those with depth 24, and 10^{-6} for those with depth 38.

b) Class Attention: On the other hand, class attention introduces a new way to handle the `[class]` token: instead of prepending it to the sequence of the input tokens at the beginning of the sequence of transformer blocks, Touvron *et al.* [31] first process the input tokens through a series of Transformer blocks (according to Eq. (7)), in a stage called *self-attention* stage. They then prepend the `[class]` token, and go through a series of blocks composed of a Multi-Head Class Attention block followed by an MLP block. They call this stage *class-attention* stage. A Multi-Head Class Attention block is like a Multi-Head Self-Attention block where only the Attention of the `[class]` to the other tokens is computed, and the other tokens are left untouched. Formally, given a `[class]` token $\boldsymbol{x}_{\text{class}}$, a vector $\boldsymbol{z} = [\boldsymbol{x}_{\text{class}}; \boldsymbol{x}_{\text{patches}}]$ given by the concatenation of the class token and the patches, learnable weight matrices $\boldsymbol{W}_q$, $\boldsymbol{W}_k$, $\boldsymbol{W}_v$, and $\boldsymbol{W}_o$ in $\mathbb{R}^{d\times d}$, and corresponding bias vectors $\boldsymbol{b}_q$, $\boldsymbol{b}_k$, $\boldsymbol{b}_v$, and $\boldsymbol{b}_o$ in $\mathbb{R}^d$ where d is the size of the token embeddings, they first perform the projections:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{W}_q \boldsymbol{x}_{\text{class}} + \boldsymbol{b}_q \\ \boldsymbol{K} &= \boldsymbol{W}_k \boldsymbol{z} + \boldsymbol{b}_k \\ \boldsymbol{V} &= \boldsymbol{W}_v \boldsymbol{z} + \boldsymbol{b}_v. \end{aligned} \tag{8}$$

They then compute class attention as

$$\text{CA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{W}_o \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d/h}}\right)\boldsymbol{V} + \boldsymbol{b}_o, \tag{9}$$

where $\boldsymbol{Q}\boldsymbol{K}^T \in \mathbb{R}^{h\times 1\times p}$, and h is the number of heads and p is the number of patches. The class-attention stage is composed of two Class Attention blocks, and the resulting architecture is dubbed *Class Attention in Transformers* (CaiT).

2) Cross-Covariance Attention and XCiT: Dosovitskiy *et al.* [18] show that using a smaller patch size brings better results. However, the attention operation has $\mathcal{O}(n^2)$ complexity for both memory and runtime, making decreasing the patch size hard. For this reason, El-Nouby *et al.* [28] propose an alternative to the attention operation, called *Cross-Covariance Attention*, with complexity $\mathcal{O}(n)$. The corresponding ViT-like architecture is called *Cross-Covariance Image Transformer* (XCiT). Overall, XCiT has a structure analogous to that of CaiT (i.e., with self-attention and class-attention phases), with the difference that it employs Cross-Covariance Attention instead of Self-Attention. As a further difference, models with depth 12 initialize LayerScale's ε to 1 instead of 0.1 (as discussed in Paragraph B1a).

a) Cross-Covariance Attention: Cross-Covariance Attention (XCA) is an attention mechanism based on cross-covariance, which works along the features dimension, i.e., along each dimension of the token embeddings. Given a queries matrix $\boldsymbol{Q}$, a keys matrix $\boldsymbol{K}$, and a values matrix $\boldsymbol{V}$, cross-covariance attention is defined as

$$\text{XC-Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\text{Softmax}\left(\frac{\hat{\boldsymbol{K}}^T\hat{\boldsymbol{Q}}}{\tau}\right), \quad (10)$$

where $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{Q}}$ are the L^2-normalized versions (i.e., with unit L^2 norm) of $\boldsymbol{K}$ and $\boldsymbol{Q}$. It is called cross-covariance attention as, in the case of self-attention $\hat{\boldsymbol{K}}^T\hat{\boldsymbol{Q}} = W_k^T X^T X W_q$ is the cross covariance matrix of $\hat{\boldsymbol{K}}$ and $\hat{\boldsymbol{Q}}$, $\text{Cov}(\hat{\boldsymbol{K}}, \hat{\boldsymbol{Q}})$. Cross-covariance is linear in time in the number of elements in X, i.e., the number of patches N. We can interpret XCA as a dynamic, data-dependent, 1×1 convolution along the axis of the features of the embeddings, as each patch is multiplied by the same data-dependent weight-matrix.

Finally, τ corresponds to a learnable temperature scaling parameter, which is applied to help the convergence of the training procedure.

b) Local Patch Interaction: Given the nature of XCA, the patches do not explicitly interact with each other. For this reason, after computing XCA, El-Nouby *et al.* [28] apply the so-called *Local Patch Interaction* (LPI), which consists of two 3×3 depth-wise convolutional layers with batch normalization and GELU activation between the two layers.

c) Convolutional Patch Projection: Differently than the previous works about ViTs introduced above, following Graham *et al.* [33], El-Nouby *et al.* [28] embed the input patches into tokens using a series of 3×3 convolutions of stride 2 with GELU activation in between. As an example, for a model with embedding dimension d_{model} with patch size 16, an RGB input image of size $(3, 224, 224)$ goes through the following transformations: $(3, 224, 224) \rightarrow (d_{\text{model}}/8, 112, 112) \rightarrow (d_{\text{model}}/4, 56, 56) \rightarrow (d_{\text{model}}/2, 28, 28) \rightarrow (d_{\text{model}}, 14, 14)$. We note that $224/16 = 14$, i.e., the final result, as expected, is a set of 14×14 vectors of size d_{model}, each of which is mapped from a patch. Finally, they use fixed, sinusoidal positional encoding as in the original work from Vaswani *et al.* [39].

C. Data augmentations

Apart from classic data augmentation strategies, such as random flipping, there are more advanced data augmentation techniques. The ones employed by Touvron *et al.* [30] are *MixUp*, *CutMix*, *RandAugment*, and *Random Erasing*.

MixUp and CutMix. MixUp [46] consists of creating an image $\tilde{\boldsymbol{X}} \in \{0, 1\}^{H \times W}$ of size (H, W) and a corresponding label $\tilde{\boldsymbol{y}}$ as the convex combination of two images and their respective labels. This means that, given two images $\boldsymbol{X}_1$ and $\boldsymbol{X}_2$, with respective one-hot-encoded labels $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$, MixUp generates an image $\tilde{\boldsymbol{X}} = \lambda \boldsymbol{X}_1 + (1 - \lambda)\boldsymbol{X}_2$, and the same is applied to the one-hot-encoded labels: the resulting label is $\tilde{\boldsymbol{y}} = \lambda \boldsymbol{y}_1 + (1 - \lambda)\boldsymbol{y}_2$. CutMix [44] follows a similar principle by cutting a portion of an image and superimposing it on another image. Formally, the resulting image is computed as $\tilde{\boldsymbol{X}} = \boldsymbol{M} \odot \boldsymbol{X}_1 + (\boldsymbol{1} - \boldsymbol{M}) \odot \boldsymbol{X}_2$, where $\boldsymbol{1}$ is the matrix of all ones, and $\boldsymbol{M}$ is a masking matrix. In particular, the masking matrix $\boldsymbol{M}$ has zeros everywhere apart from the bounding box $\boldsymbol{B}$ delimited by the coordinates (r_x, r_y, r_h, r_w), where r_x and r_y are sampled uniformly along the height and the width of the image, $r_h = H\sqrt{1 - \lambda}$ and $r_w = W\sqrt{1 - \lambda}$. In this way, the box is placed randomly in the image and has area proportional to λ. We show some examples for these data augmentations in Fig. 9.

(a) MixUp example.

(b) CutMix example.

Fig. 9: Examples for MixUp *(Top)* and CutMix *(Bottom)* data augmentations.

RandAugment. RandAugment [45] improves the so-called *automated augmentations*, which automatically select the best augmentations for a given model and task, among a given list of possible transformations (e.g., rotation and brightness change). Automated augmentations are effective, but need a separate search phase. RandAugment reduces the search space, which enables training without a prior search phase. In particular, given K augmentations, RandAugment chooses each transformation with probability $1/K$. We show an example for three RandAugment augmentations in Fig. 10.

Random Erasing. Finally, Random Erasing [47] randomly selects a portion of pixels in an image, and occludes them, either by setting them to 0 or by sampling their value from a normal distribution with mean and standard deviation equal to those of the dataset. We show an example for Random Erasing in Fig. 11.

D. Setup and hyperparameters

In this section, we give additional details about the training, evaluation, and implementation setups.

Training hyperparameters. Apart from the experiments on larger models, we run all the training runs using VMs with 8 TPUv3 cores. In all the runs, unless otherwise stated, we use the same setup as the one used for DeiT [30]. We use as a batch size $64 \times 8 = 512$ (i.e. 64 samples per TPU core), and the learning rate is chosen according to the formula provided by Touvron *et al.* [30] (i.e. $\text{lr} = 0.0005 \times \frac{\text{batch size}}{512}$), which corresponds to 0.0005 with the batch size we use in most experiments. For all the training runs on ImageNet-1k, we train the model for 110

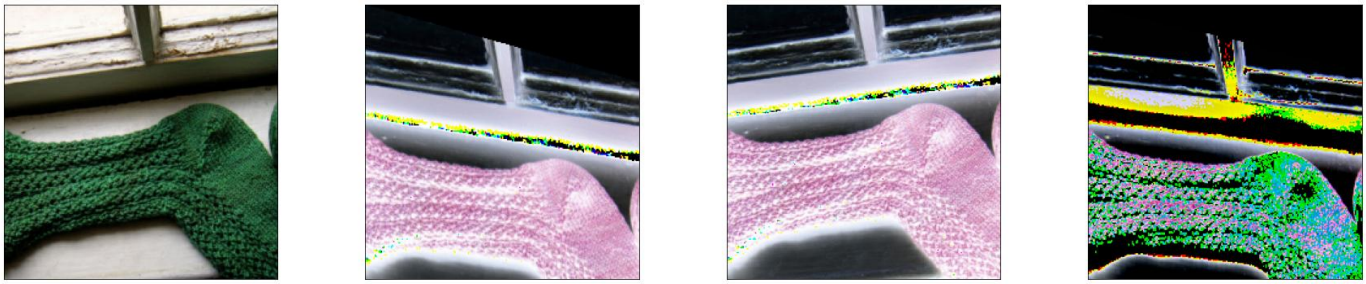


Fig. 10: Original image *(Left)* and three examples of augmented images generated by RandAugment

Fig. 11: Original image *(Left)* and example of erasure generated by Random Erasing *(Right)*.

epochs, with a learning rate cosine decay with a final value of 5×10^{-5}, a 10-epochs warm-up from 5×10^{-6} and 10 epochs cool-down. We use the AdamW optimizer [75]. Apart from the architecture ablation (Section III-B), we do not employ *repeated augmentations* [76] to save training time. Repeated augmentations consist of repeating each batch 3 times: the first one without data augmentations and the following ones with it. Hence, employing repeated augmentations would be equivalent to doing $3\times$ the number of epochs. On the other hand, for XCiT-M12, XCiT-L12, ConvNeXt, and PoolFormer, we use TPUv4 pods with either 32 or 64 TPU cores. As mentioned above, we increase the total batch size according to the model size and on the number of devices in use, and we scale the learning rate according to the rule stated above.

Training runtime. We train all three XCiT variants on a pod with 64 TPUv4 cores to compare the training time. We use the largest batch size that can fit into each device, which is 256 for XCiT-S12 and XCiT-M12, and 128 for XCiT-L12. We scale the learning rates as described in the paragraph above. The total training time for XCiT-S12 is 19h30m, for XCiT-M12 it is 33h, and for XCiT-L12 it is 39h.

Training attack setup. Finally, unless otherwise stated, we use FGSM for adversarial training [54], initializing the adversarial perturbation to be uniformly distributed in $[-\varepsilon, \varepsilon]$, and adding 10^{-5} to avoid numerical instability. Moreover, we apply early-stopping, i.e., we evaluate the checkpoint at the epoch where the model was performing best in terms of FGSM accuracy on the test set.

Large epsilon pre-training setup. We pre-train an XCiT-S12 on ImageNet-1k with $\varepsilon = {}^{8}/_{255}$. Using the same setup as the $\varepsilon = {}^{4}/_{255}$ training, we observe strong label leaking (which was also observed by previous work on ViTs adversarial training [77]). To solve this, we use 2-steps FGSM instead of 1-step FGSM as the attack for adversarial training. By doing so, we manage to train a model which has 25.00% AutoAttack accuracy and 63.46% clean accuracy on the subset of 5000 images from RobustBench when using $\varepsilon = {}^{8}/_{255}$ as attack budget. We similarly train an XCiT-M12 and an XCiT-L12, using the same setup. Moreover, as a baseline, we attempt to pre-train an XCiT-S12 robust to $\varepsilon = {}^{8}/_{255}$ perturbations using the canonical training recipe. However, we observe that the training fails. For this reason, we adopt the epsilon warm-up from our tailored training recipe. Finally, we pre-train a ResNet-50 with GELU activation function, using the same setup as Bai *et al.* [2]. To validate the correctness of our implementation and setup, we first successfully reproduce their results with $\varepsilon = {}^{4}/_{255}$. However, when training a model robust to $\varepsilon = {}^{8}/_{255}$ perturbations with this setup, the resulting model is worse than the ReLU ResNet-50 pre-trained by Salman *et al.* [50][5]. For this reason, in the fine-tuning experiments, we fine-tune this network. We show the full pre-training results in Table XI.

High-resolution finetuning setup. We fine-tune the model pre-trained on ImageNet-1k with $\varepsilon = {}^{8}/_{255}$ on the high-resolution datasets Caltech-101 and Oxford Flowers. We fine-tune using $\varepsilon = {}^{8}/_{255}$ for 20 epochs and the same training recipe as the one used for pre-training, with the difference that we do adversarial training with 1-step FGSM instead of 2-steps, and we do not employ a warm-up for ε. We also do a fine-tuning run without adversarial training to better quantify the clean-robust accuracy tradeoff.

Low-resolution finetuning setup. The pre-trained XCiT models are meant for inputs with patch size 16. However, such a patch size would be too large for datasets of smaller images such as CIFAR-10 and CIFAR-100 (which have 32×32 resolution). For this reason, we need a way to adapt the model to support a different patch size. Previous work [23] achieves this by down-sampling the weights of the convolutional layer that is used by ViT to embed each patch. However, XCiT uses 4 subsequent convolutional layers to embed 16×16 patches into

[5]The checkpoints from this paper can be downloaded from https://github.com/microsoft/robust-models-transfer

1-D vectors, and each layer has stride 2 [33]. To embed 4×4 patches, we need to use just 2 subsequent convolutions with stride 2. For this reason, we adapt our model by setting the stride of the first two convolutional layers to 1. Regarding the ResNet-50, instead, we substitute the first convolutional layer (which has originally kernel size 7 and stride 2) with one with kernel size 3 and stride 1, and we remove the first pooling layer. Given the smaller size and resolution of the CIFAR datasets, we fine-tune the robustly trained model using TRADES [10], with PGD-10 as the attack. Similar to the high-resolution datasets, we fine-tune for 20 epochs. However, we remove the color jitter data augmentation, as the inputs are smaller and would make the task too hard. Moreover, we search for the best learning rate, which we find to be 2×10^{-4} (as opposed to 5.0×10^{-5} that we used for pre-training and the high-resolution datasets). We probably need a larger learning rate to better tune the input embedding layer whose structure we change. Finally, given the smaller resolution of the images, we change the values for the random scale and crop data augmentation as follows: the ratio of possible crop ranges from $[0.75, 1.33]$ to $[0.95, 1.05]$, and the input re-scaling range from $[0.08, 1.0]$ to $[0.8, 1.2]$. If we kept these large ranges, then very few pixels of the original image would remain after cropping and resizing, hence the task would be too hard, and the model would underfit.

Evaluation setup. For the final ablation and the scaled-up models trained on ImageNet-1k, we run AutoAttack [38], an ensemble of white- and black-box attacks. We run the attack on the subset of 5000 ImageNet-1k images used for the RobustBench benchmark [24]. Given that AutoAttack is composed of four attacks, two of which are black-box, it is computational expensive. To strike a balance between strength and computational cost, instead, we assess the robustness of the individual ablations using APGD-CE [38]. APGD-CE is a parameter-free attack, which is the first of the ensemble that makes up AutoAttack. We run this attack with 5 restarts and 100 iterations, the same settings of the attack that is part of AutoAttack. Finally, we compute the FLOPs and number of parameters using the `fvcore` library[6].

Additional implementation details. We base our implementation on the PyTorch Image Models repository [60][7], which includes the `timm` library and provides a template training script. This library uses the PyTorch framework [43]. In particular, given that we run our experiments on Tensor Processing Unit (TPU) devices, we use the PyTorch XLA library, which compiles PyTorch code to the XLA[8] Intermediate Representation (XLA IR), needed to run the computations on TPUs. The XLA IR consists of a graph representing the computation performed on tensors. The graph is then compiled and optimized (e.g., by fusing operations when possible). To use `timm`'s utility functions to work with TPUs, we use the `bits_and_tpu` branch of PyTorch Image Models[9], which introduces the compatibility of the library with XLA and TPUs. We adapt `timm`'s default training script to perform adversarial training.

Because of an existing bug in PyTorch XLA[10], when we run the attacks (e.g., FGSM) during training, we have to set the model to the `.train()` mode (which influences the behavior of batch normalization layers). However, this should not impact the overall robustness of the models [48]. Moreover, while all the AutoAttack and APGD-CE evaluations are run on V100 GPUs, hence with the model in `.eval()` mode, in the case of the attack effectiveness experiment (sec:attack-effectiveness) we run the evaluations on TPUs, thus with the model in `.train()` mode. We do so as the large number of evaluations we run (160) would have been unfeasible with our GPU compute budget.

Carbon emissions. Finally, the carbon footprint of the project, measured via Google Cloud's Carbon Footprint Console[11], is 33 $kgCO_2$. For scale, a flight from Paris to London generates around 55.7 $kgCO_2$ per person in economy class[12].

E. Additional ablation results

We show in Table IX the full results for the data augmentation ablation. We can observe that the setups with the heaviest data augmentations rank at the bottom.

F. PGD Results

We report the results for PGD attacks in Table X.

G. Pre-training results

We show in Table XI the results for training runs with $\varepsilon = {}^{8}/_{255}$. These are the models we use for fine-tuning.

H. Additional plots regarding attack effectiveness

We show, in Figs. 12 and 13 additional results regarding the attack effectiveness experiment.

I. Additional samples for the optimization sanity checks

We show the loss for different attack steps for 32 different samples from ImageNet-1k in Fig. 14 (XCiT-S12), and Fig. 15 (ResNet-50), and the loss progression of the loss in one attack targeting the same 32 samples in Fig. 16 (XCiT-S12), and Fig. 17 (ResNet-50).

[6] https://github.com/facebookresearch/fvcore

[7] https://github.com/rwightman/pytorch-image-models/

[8] XLA stands for Accelerated Linear Algebra, more information about the XLA compiler can be found here: https://www.tensorflow.org/xla/architecture

[9] https://github.com/rwightman/pytorch-image-models/tree/bits_and_tpu. The branch may be eventually merged into `main`. Hence, this URL may become invalid.

[10] https://github.com/pytorch/xla/issues/3361

[11] https://cloud.google.com/carbon-footprint

[12] Computed on https://www.icao.int/environmental-protection/Carbonoffset/Pages/default.aspx

TABLE IX: **Weak data augmentation is better.** The strategies that perform best are those with just Random Erasing, or no heavy augmentation, such as RandAugment, at all. We report the full results results in this table, sorted by APGD-CE accuracy. In all the runs we keep weak data augmentation that are commonly used (random flip and crop, and color jitter). (Arch: XCiT-N12)

Data Augmentation Policy				Accuracy	
MixUp	CutMix	RandAugment	Random Erasing	*Clean*	*APGD-CE*
✗	✗	✗	✓	**67.28**	**39.22**
✗	✗	✗	✗	66.78	**39.22**
✓	✗	✗	✗	61.04	38.56
✓	✗	✗	✓	60.46	38.26
✓	✓	✗	✗	62.04	38.18
✗	✗	✓	✗	65.34	37.64
✗	✗	✓	✓	64.76	37.62
✓	✓	✗	✓	59.80	37.20
✓	✗	✓	✗	57.16	36.74
✗	✓	✗	✗	61.62	36.30
✓	✓	✓	✗	57.60	36.06
✗	✓	✗	✓	61.70	35.74
✓	✗	✓	✓	55.64	35.70
✓	✓	✓	✓	55.96	35.38
✗	✓	✓	✗	56.64	32.92
✗	✓	✓	✓	55.64	32.40

TABLE X: **PGD accuracy decreases with the number of steps.** We run the PGD attack with 5, 10, 50, and 100 steps for $\varepsilon = 8/255$. We observe that, as expected, the robust accuracy gently plateaus at 50 steps, with no significant difference between 50 and 100 steps, for all three models. We run the attack on the full ImageNet validation set, using as a step size $1.5 \cdot \varepsilon/n$), where n is the number of attack steps.

Model	Clean Accuracy	Robust Accuracy			
		PGD-5	PGD-10	PGD-50	PGD-100
XCiT-S12	72.34	49.16	48.91	48.71	48.69
XCiT-M12	74.04	51.96	51.71	51.55	51.53
XCiT-L12	73.76	53.75	53.52	53.37	53.36

TABLE XI: **The training recipe also works for larger epsilons.** Results for training with $\varepsilon = 8/255$: we can observe that, for XCiT, the performance improves with scale.

Model	Clean Accuracy	AA Accuracy
GELU ResNet-50	58.08	17.14
ReLU ResNet-50 [50]	54.90	19.72
XCiT-S12	63.46	25.00
XCiT-M12	67.80	26.58
XCiT-L12	69.24	28.74

J. Additional experiment about XCiT's gradients

Direct feature visualization. Gradients of robust CNNs are more aligned with human perception [35]. In particular, in their work, they introduce a visualization technique called direct feature visualization, by which they maximize the output of a model at a specific activation in the penultimate layer by optimizing an input image via PGD. They observe that the images generated in this way using an adversarially-trained model contain semantically meaningful information without the need for regularization terms on the input. We explore a variation of this experiment: instead of maximizing a specific activation, starting from uniformly random inputs, we run a targeted attack that targets a random class, i.e., we change the input so that it is classified with the given class with the highest confidence. We do so by optimizing the input via PGD-100, using $\varepsilon = 15$. To the best of our knowledge, we are the first to run a similar experiment on a robust ViT-like model. We can see a set of random images and classes in Fig. 18.

We make the following observations: 1) The non-robust XCiT gradients have no semantic meaning. 2) Regarding the first image on the left, whose target class is "small white" (a butterfly species), for both the robust models, we can see a

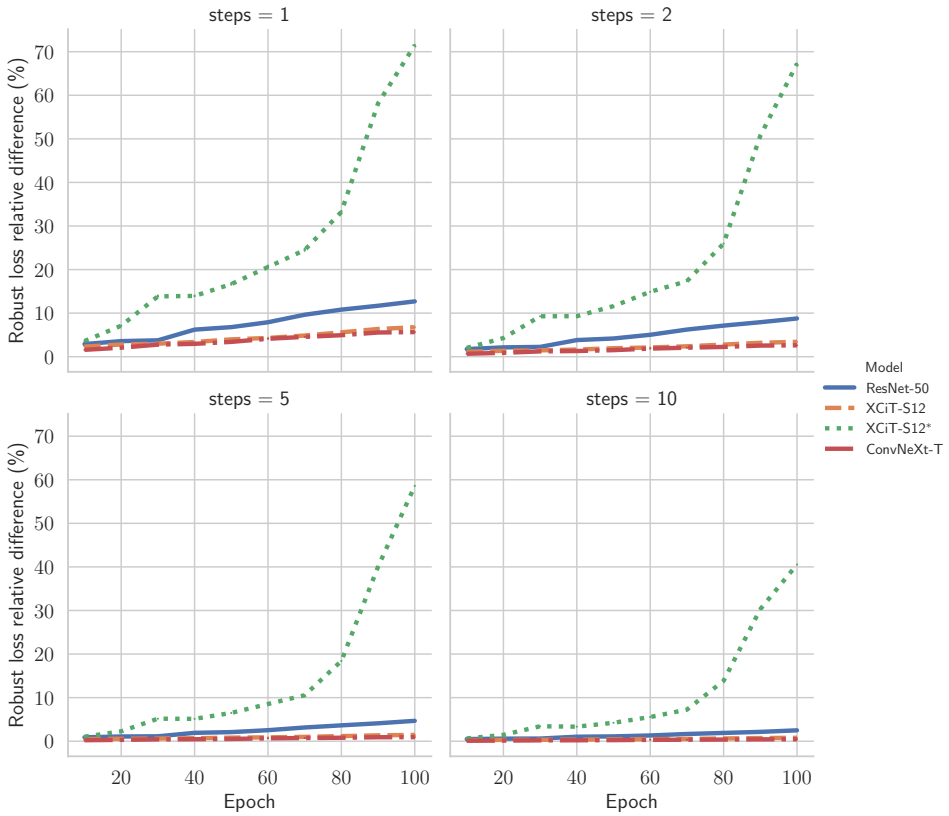


Fig. 12: Comparison, across different attack steps, of the relative difference between the adversarial loss computed with the given attack steps and the adversarial loss computed with PGD-200, and how this quantity changes every 10 training epochs.

white portion in the shape of a butterfly with a black spot, which is what a small white butterfly looks. 3) Regarding the image targeting "feather boa" (feathery party apparel), we can see, in the case of the robust XCiT-S12, long, colorful structures with feather-like edges. 4) For the images targeting the "pot" class, we can see the borders of a pot in the case of the robust XCiT-S12. We can also observe that we can see some plants as well, meaning that, probably, in the datasets, pots are most often represented when containing plants. 5) Finally, in the last image on the right, which should target the "border collie" class, we can see a Border Collie for the robust XCiT-S12 and the head of one in the lower right corner for the robust ResNet-50.

Perturbations visualization. We also visualize the adversarial perturbations generated with a PGD-100 attack for a robust and a non-robust XCiT, compared to those generated for a robust ResNet-50. Given that a perturbation $\boldsymbol{\delta}$ is in $[-\varepsilon, \varepsilon]$, we rescale it to $[0, 1]$ to visualize it as an image. For this reason, we compute the visualized images $\delta_{\text{viz}} = \frac{\delta+\varepsilon}{2\varepsilon}$ and we visualize the intensity of the perturbation by transforming the image to grey-scale colors.

We can see a random sample of images and their respective perturbations in Fig. 8. We note that the shapes of the original images are visible in the robust XCiT and ResNet perturbations, while they are not in the non-robust ones.

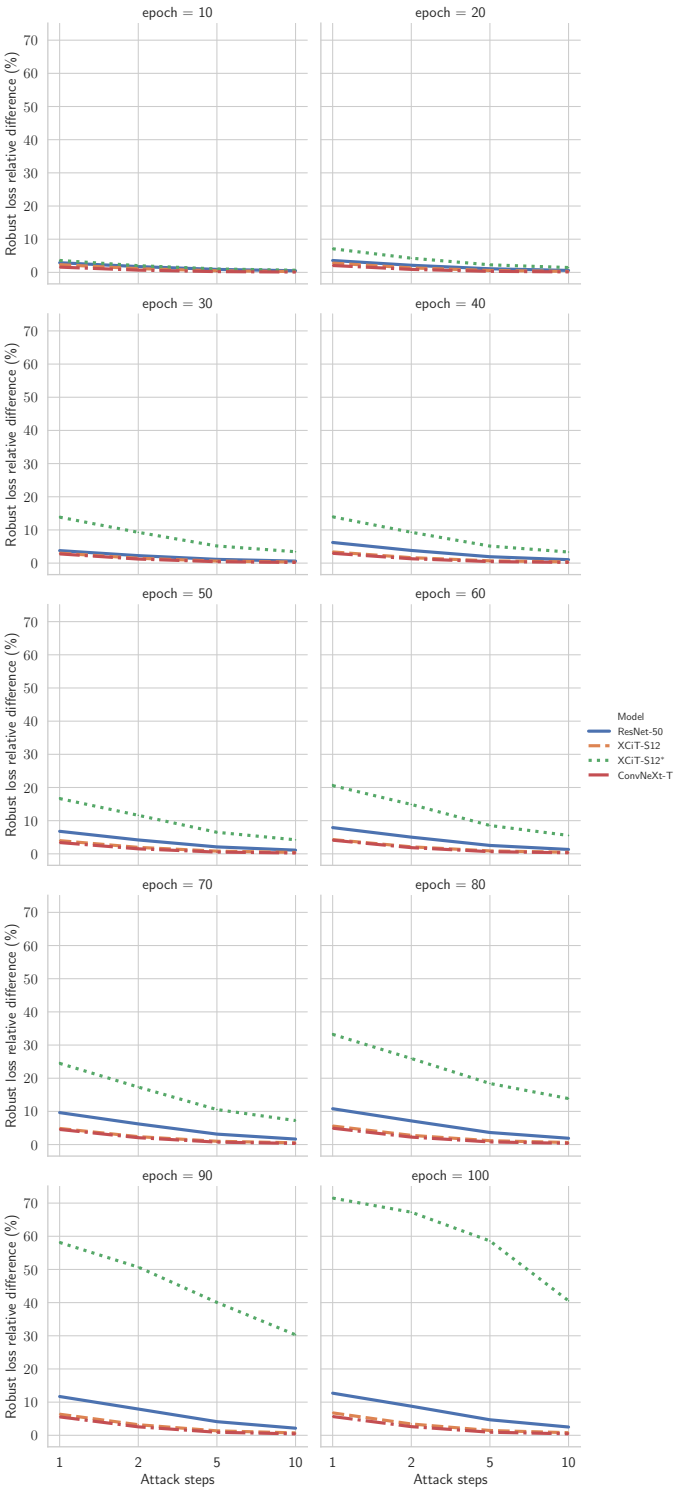


Fig. 13: Comparison, every 10 epochs, of the relative difference between the adversarial loss computed with different numbers of attack steps and the adversarial loss computed with PGD-200.

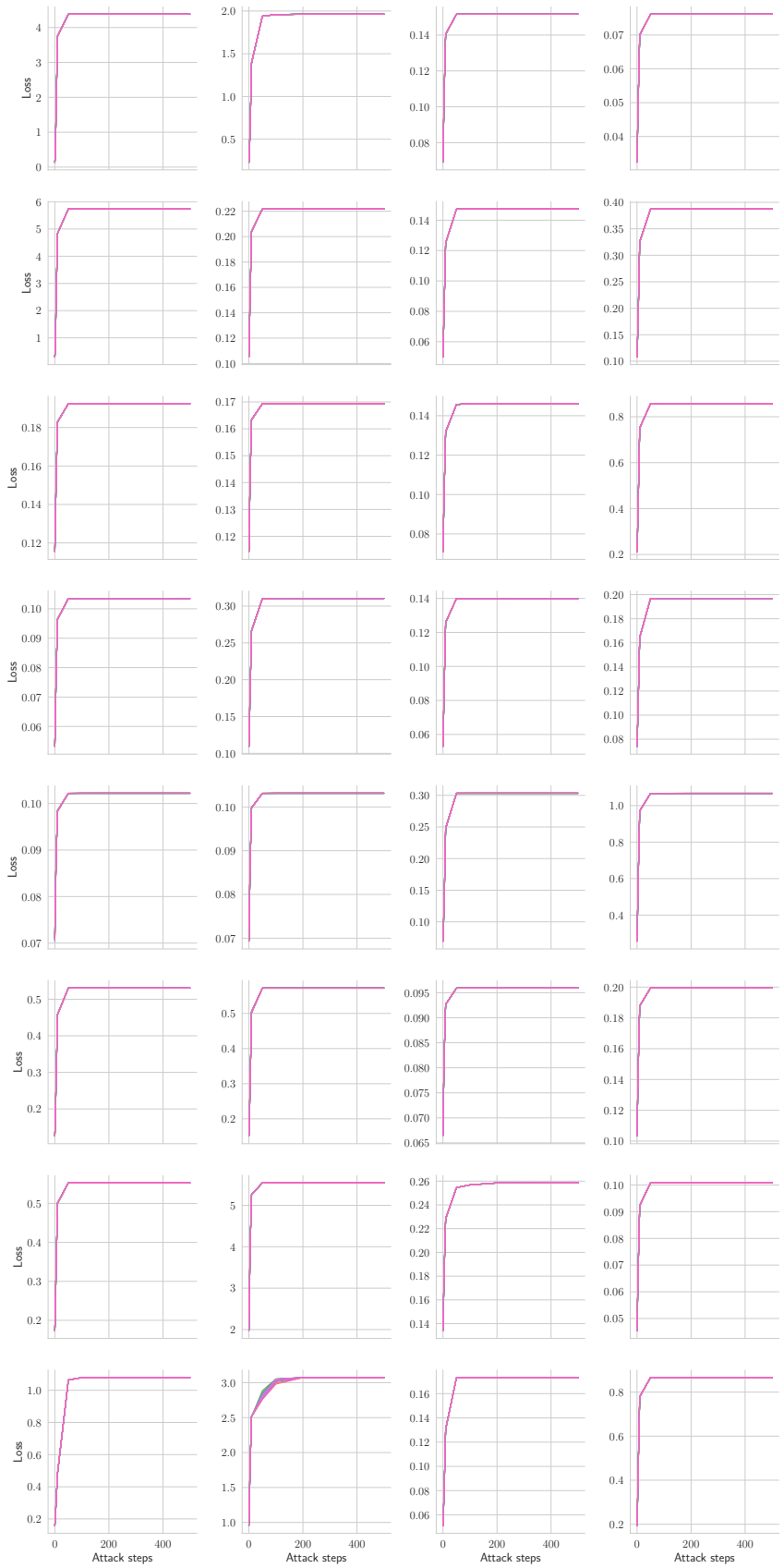


Fig. 14: Comparison between different runs of PGD attacks with different numbers of steps for XCiT-S12, for 32 different random points from ImageNet-1k.

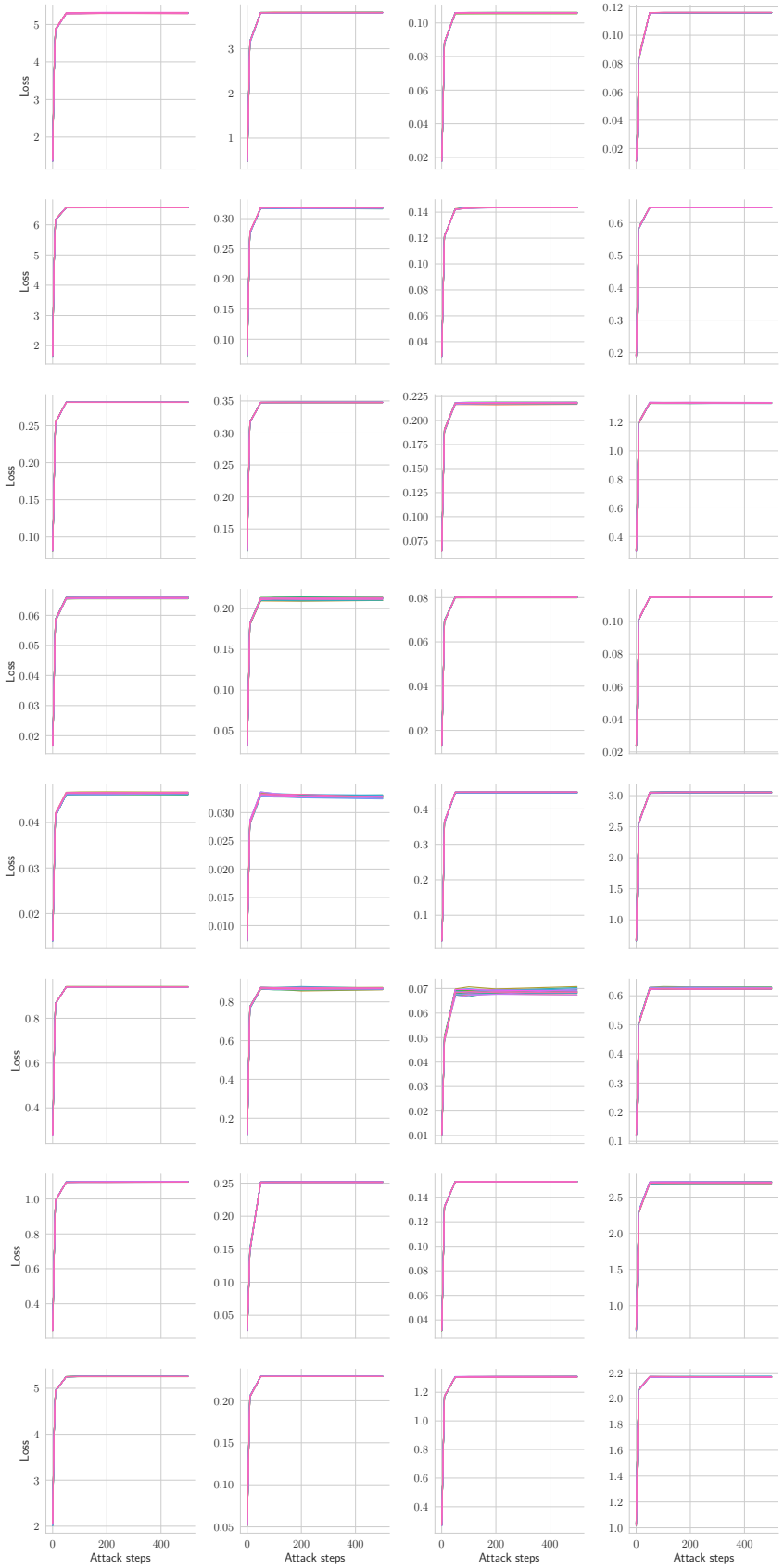


Fig. 15: Comparison between different runs of PGD attacks with different numbers of steps for GELU ResNet-50, for 32 different random points from ImageNet-1k.

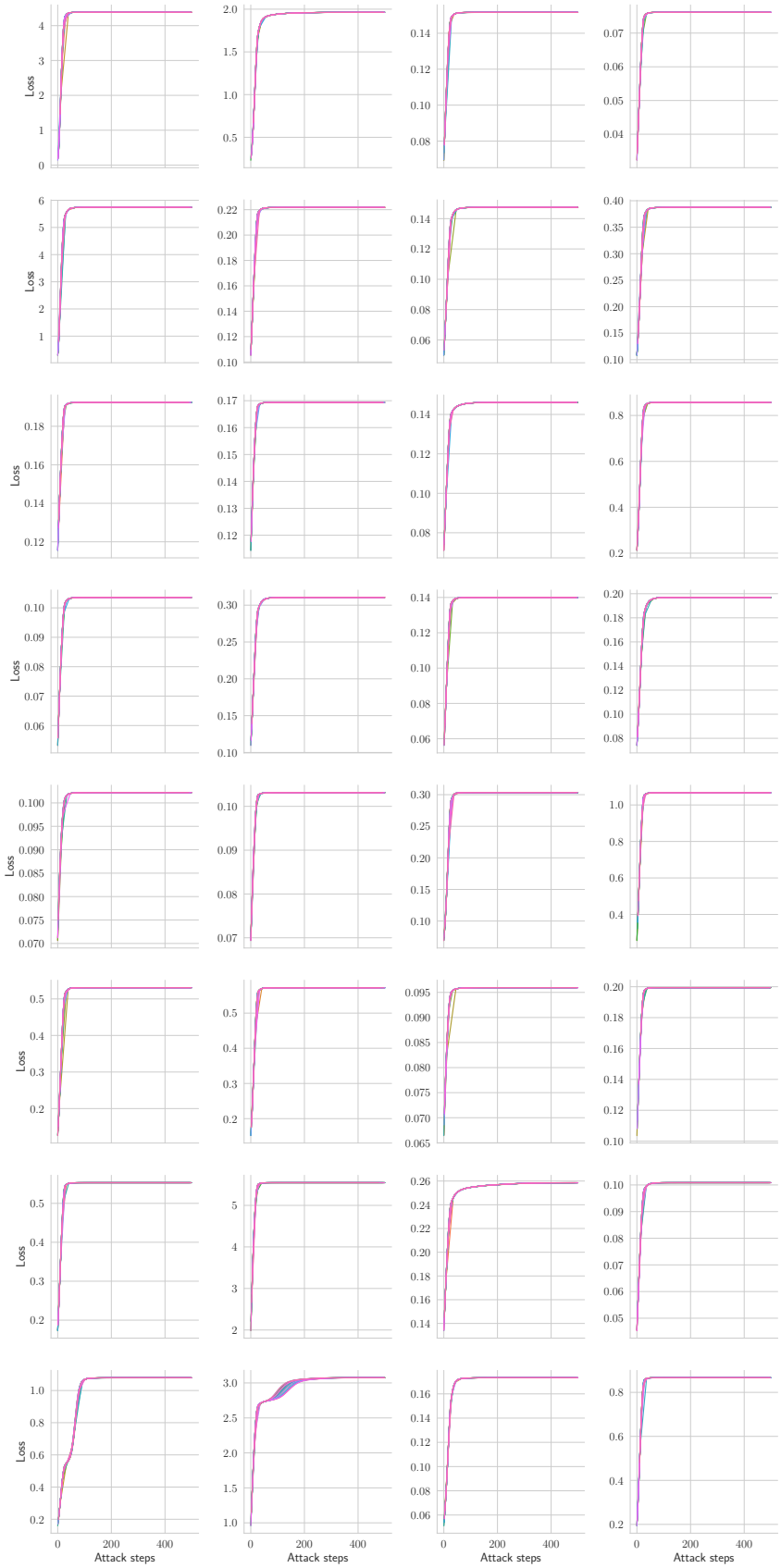


Fig. 16: Evolution of the loss for different runs of an attack, using a large step size for XCiT-S12, for 32 different random points from ImageNet-1k.

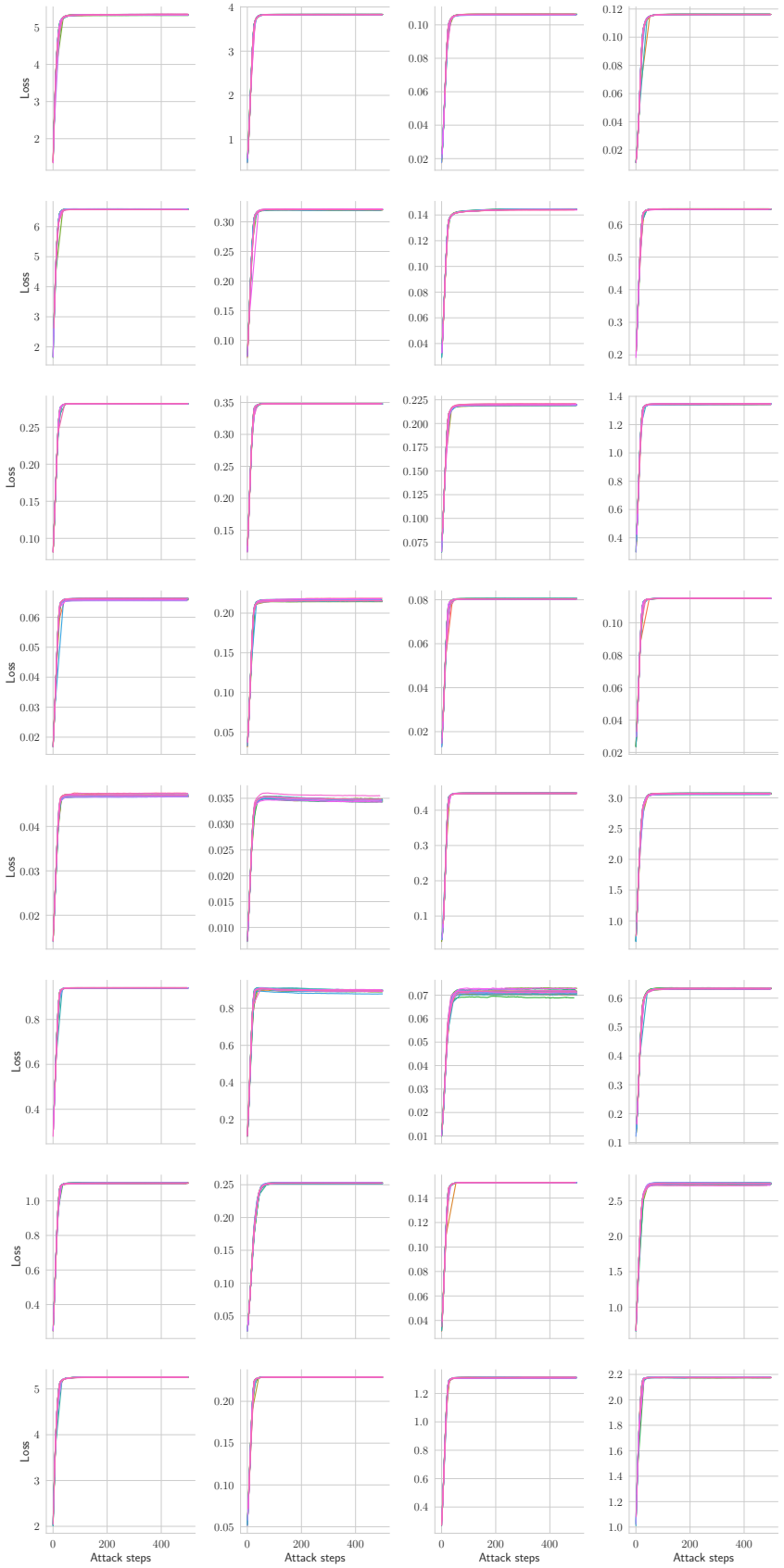


Fig. 17: Evolution of the loss for different runs of an attack, using a large step size for GELU ResNet-50, for 32 different random points from ImageNet-1k.

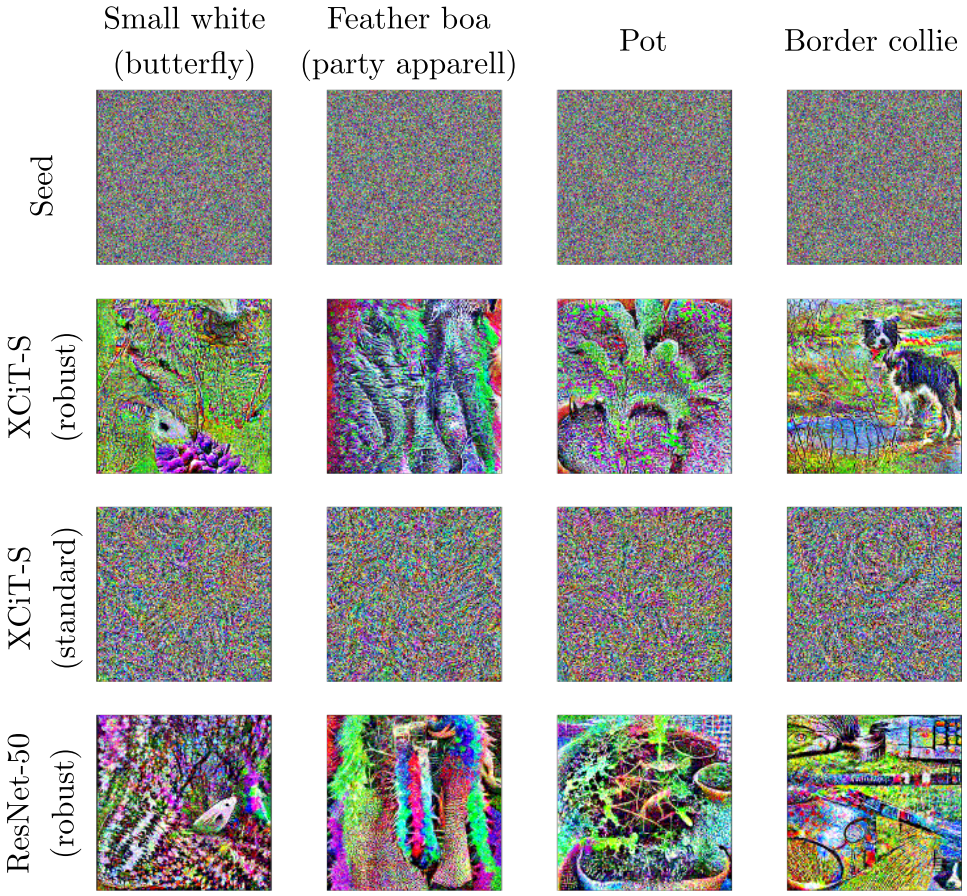


Fig. 18: Comparison between the gradient accumulation for a robust XCiT-S12 and a non-robust ResNet-50.