Adversarially Robust Vision Transformers

Master's Thesis of

Edoardo Debenedetti
Department of Information and Communication Sciences (IC)
Swiss Federal Institute of Technology, Lausanne (EPFL)

Supervised by
Vikash Sehwag and Prateek Mittal — Princeton University
Carmela Troncoso — EPFL

October-April 2022

To my parents, who helped me equip the boat.
To my sister, who set the best example on how to sail.
To Anita, who is sailing with me in this exciting journey.

Acknowledgments

First and foremost, I would like to thank Vikash, who, during our meetings, has always given me extremely constructive feedback about the ideas I proposed and I wanted to discuss. With his expertise, he gave me great suggestions and pointed me to useful resources. He also shared essential insights not only about adversarial robustness but also about research in general, as well as writing and presenting this work. Next, I would like to thank Maksym, with whom I had interesting and fruitful discussions during the last year of my Master's degree at EPFL. Both of them also were extremely helpful when I was preparing my Ph.D. applications, so thanks also for that!

I would also like to thank Google's TPU Research Cloud (TRC) Program[1], which provided me with extremely generous computing resources. Without such generous resources, this thesis would have been impossible, and I couldn't have explored the topic of adversarially training Vision Transformers.

Then, I would like to thank my friends Edoardo and Jacopo, with whom I (unfortunately, remotely) shared the excitements (or, from time to time, lack thereof) related to research. Finally, I would like to truly thank my parents, my sister, and Anita, who have supported me throughout my studies, even though studying brought away some of the time I could have otherwise spent with them.

[1] `https://sites.research.google/trc/about/`

Notation

In this thesis, we will use the following notation conventions, borrowed from I. Goodfellow et al. (2016)[2].

	Numbers and Arrays
a	A scalar (integer or real)
$\boldsymbol{a}$	A vector
$\boldsymbol{A}$	A matrix
$\text{diag}(\boldsymbol{a})$	A square, diagonal matrix with diagonal entries given by $\boldsymbol{a}$
a	A scalar random variable
$\mathbf{a}$	A vector-valued random variable
$\mathbf{A}$	A matrix-valued random variable
$A_{i,j}$	Element (i, j) of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{i,:}$	Row i of matrix $\boldsymbol{A}$
$\boldsymbol{A}_{:,i}$	Column i of matrix $\boldsymbol{A}$
	Sets
$\mathbb{A}$	A set
$\mathbb{R}$	The set of real numbers
$[a, b]$	The real interval including a and b
$(a, b]$	The real interval excluding a but including b
	Linear Algebra Operations
$\boldsymbol{A}^\top$	Transpose of matrix $\boldsymbol{A}$
$\boldsymbol{A} \odot \boldsymbol{B}$	Element-wise (Hadamard) product of $\boldsymbol{A}$ and $\boldsymbol{B}$
	Calculus
$\frac{dy}{dx}$	Derivative of y with respect to x
$\frac{\partial y}{\partial x}$	Partial derivative of y with respect to x
$\nabla_{\boldsymbol{x}} y$	Gradient of y with respect to $\boldsymbol{x}$
$\nabla_{\boldsymbol{X}} y$	Matrix derivatives of y with respect to $\boldsymbol{X}$

[2]A TeX file with the notation can be downloaded at `https://github.com/goodfeli/dlbook_notation/`.

Probability

$p(\mathrm{a})$	A probability distribution over a continuous variable, or over a variable whose type has not been specified
$\mathrm{a} \sim P$	Random variable a has distribution P
$\mathbb{E}_{\mathrm{x} \sim P}[f(x)]$ or $\mathbb{E}f(x)$	Expectation of $f(x)$ with respect to $P(\mathrm{x})$
$\mathrm{Var}(f(x))$	Variance of $f(x)$ under $P(\mathrm{x})$
$\mathrm{Cov}(f(x), g(x))$	Covariance of $f(x)$ and $g(x)$ under $P(\mathrm{x})$
$D_{\mathrm{KL}}(P \| Q)$	Kullback-Leibler divergence of P and Q

Functions

$f : \mathbb{A} \to \mathbb{B}$	The function f with domain $\mathbb{A}$ and range $\mathbb{B}$
$f \circ g$	Composition of the functions f and g
$f(\boldsymbol{x}; \boldsymbol{\theta})$	A function of $\boldsymbol{x}$ parametrized by $\boldsymbol{\theta}$. (Sometimes we write $f(\boldsymbol{x})$ and omit the argument $\boldsymbol{\theta}$ to lighten notation)
$\|\|\boldsymbol{x}\|\|_p$	L^p norm of $\boldsymbol{x}$
$\|\|\boldsymbol{x}\|\|$	L^2 norm of $\boldsymbol{x}$

Sometimes we use a function f whose argument is a scalar but apply it to a vector, matrix, or tensor: $f(\boldsymbol{x})$, $f(\boldsymbol{X})$, or $f(\mathbf{X})$. This denotes the application of f to the array element-wise. For example, if $\mathbf{C} = f(\mathbf{X})$, then $C_{i,j,k} = f(X_{i,j,k})$ for all valid values of i, j and k.

Datasets and Distributions

p_{data}	The data generating distribution
$\hat{p}_{\mathrm{data}}$	The empirical distribution defined by the training set
$\mathbb{X}$	A set of training examples
$\boldsymbol{x}^{(i)}$	The i-th example (input) from a dataset
$y^{(i)}$ or $\boldsymbol{y}^{(i)}$	The target associated with $\boldsymbol{x}^{(i)}$ for supervised learning
$\boldsymbol{X}$	The $m \times n$ matrix with input example $\boldsymbol{x}^{(i)}$ in row $\boldsymbol{X}_{i,:}$

Contents

Abstract

Machine learning models are vulnerable to adversarial examples: perturbations added to benign inputs in order to fool a model into making a wrong prediction. The most successful approach to defend against adversarial examples is adversarial training, a training technique which is theoretically principled and highly effective in practice. Adversarially trained models are more robust to adversarial perturbations, albeit at the expense of the accuracy on clean samples, leading to a robustness-accuracy trade-off. Currently, the community resorts to deeper and wider models to improve this trade-off, hence decreasing the efficiency and practicality of adversarial training. In this work we show that, by switching to Vision Transformers (in particular XCiT, a Vision Transformer variation) than the ones most commonly used (ResNets and WideResNets), we can improve this trade-off without the need to use larger models, hence improving the practicality of adversarial training. We manage to do so by finding a tailored adversarial training recipe –different from the default recipe for standard training– which leads to state-of-the-art results by a significant margin. We also show that this setup scales to larger variants of XCiT, and that models trained with this setup can be fine-tuned on other smaller datasets, such as CIFAR-10, Caltech-101, and Oxford Flowers. Moreover, we compare the adversarial perturbations of our robust XCiT to those of a robust ResNet, quantifying that the former captures more semantic attributes than the latter.

To the best of our knowledge, this is the first work to establish superiority of Vision Transformer over CNNs in robust machine learning. Thus, we highly recommend the use of ViTs for adversarial training.

1 Introduction

Despite their success, machine learning models are vulnerable to small perturbations –called *adversarial examples*– that are often imperceptible to the human eye (Biggio et al. 2013; Szegedy et al. 2013). This can be a cause of concern if the models are deployed to real-world, safety-critical systems (Kurakin et al. 2016). In the case of image classification, such perturbations can mislead the model, and make it predict a different label than the correct one. The study of the adversarial robustness for Convolutional Neural Networks (CNNs) is a well-established sub-field of the machine learning research community, and many defenses have been proposed. The large majority of the defenses are based on *adversarial training* (Madry et al. 2017), which consists of training the model on adversarial examples instead of clean data. Previous work mostly focuses on either the algorithmic aspect of training (Hongyang Zhang et al. 2019; Rade et al. 2021) or on ways to better leverage data (Rebuffi et al. 2021; Sehwag et al. 2021; Gowal et al. 2021) instead of architectural components (Huang et al. 2021). As a matter of fact, ResNet variations, such as ResNet (He et al. 2015) and WideResNet (Zagoruyko et al. 2016) are the *de facto* standard when it comes to training models robust to adversarial examples (Gowal et al. 2021; Rebuffi et al. 2021; Pang et al. 2022; Rade et al. 2021).

Notwithstanding a big effort from the research community, adversarial training hurts the accuracy of the model on clean data. Moreover, it is possible to observe in RobustBench's leaderboard (Croce et al. 2020a) that, to keep a good robustness-accuracy tradeoff, the community resorts to deeper, wider, and less efficient models, creating a *robustness-accuracy-efficiency trade-off*. Previous work also attempts to improve robustness by innovating the architectural side: the focus has been on CNNs, e.g., by trying different activation functions (Xie et al. 2020; Bai et al. 2021). In this work, we do a step toward answering the following question: can we get a better accuracy-robustness-efficiency trade-off with tools and architectures other than ResNets? Can we improve the current state of the art with the plain formulation of adversarial training without paying high costs in terms of efficiency and by leveraging different architectures such as Vision Transformers (Dosovitskiy et al. 2021)?

Adversarial training for ViTs needs custom-tailored training recipes. Vision Transformers outperform CNNs in terms of clean accuracy when standardly trained (Dosovitskiy et al. 2021). However, training them adversarially to do better than CNNs is not straightforward. The reason is that training ViTs is, in general, a nontrivial task due to their lack of inductive bias, thus

needing their own custom training recipes (Steiner et al. 2021). Nonetheless, we show that it is possible to use ViTs to bring improvements on all three fronts of the robustness-accuracy-efficiency trilemma. As the training setup is an important component of training Vision Transformers, we first explore an effective adversarial training recipe. Instead of just taking the original, vanilla training recipe used by ViTs and successive works (Steiner et al. 2021; Touvron et al. 2021a), 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 original 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 (Steiner et al. 2021; Touvron et al. 2021a), this is not the case for adversarial training.

Using this tailored recipe, we show that we can effectively adversarially train a ViT-like architecture (XCiT, El-Nouby et al. (2021)), in a variant comparable to ResNet-50 in terms of the number of parameters and FLOPs. With our setup, this variant improves both robust and clean accuracy by a significant margin relative to the well-trained ResNet-50 by Bai et al. (2021) (by 17.7% and 7.3% respectively). We then show that this setup successfully scales up to larger variants of XCiT, which achieve even better results (74.04% clean accuracy and 45.24% AutoAttack accuracy). Given the staggering difference, we highly recommend that researchers in the field should use Vision Transformers in adversarial training.

Adversarially trained XCiTs can be leveraged for fine-tuning. Further, given the recipe we identify, we answer the following question: can we efficiently leverage the pre-trained models by doing adversarial fine-tuning, keeping the same good trade-off, thus achieving a better tradeoff than CNNs on smaller datasets too? We show that this setup also works for larger perturbations with minimal changes and that the resulting model can be successfully fine-tuned on datasets in the low-data regime of both coarse- and fine-grained nature (e.g., Caltech-101 (Fei-Fei et al. 2004) and Oxford Flowers (Nilsback et al. 2008)). Also for this case, we employ XCiT, which gives a very good trade-off, with just a 1% drop in clean accuracy between adversarial and non-adversarial fine-tuning in the case of Caltech-101.

Gradients of adversarially robust XCiTs. Given the different nature of ViTs, we finally do a brief analysis of the behavior of an adversarially

trained XCiT, by studying its gradients. We first visualize the adversarial perturbations of both the robust XCiT, as well as those of a robust ResNet-50. Using quantitative metrics, we then show that the former ones are more semantic than the latter ones. We then visualize the semantic nature of the gradients by accumulating gradients to maximize specific, randomly selected classes (Engstrom et al. 2019) and show that we get high-quality visualizations that are semantically meaningful to the human eye.

Finally, we share[3] the checkpoints of our models –including the optimizer states– for different sizes and perturbations to enable other researchers to fine-tune the models, as well as run further analyses.

This thesis is structured as follows: we first cover the background by discussing the basics of adversarial robustness and adversarial training (section 2.1), and an introduction to ResNets (section 2.2), the Transformer architecture and some of its later developments (section 2.3). We then cover related work about the adversarial robustness of vision transformers (section 3.2). Next, we present our experiments about adversarially training ViT-like models (section 4), and we finally conclude with a discussion of potential future work (section 5).

[3]The checkpoints are on Google Drive (`https://drive.google.com/drive/folders/1Q_E3ryzgLCkvrtmrUjal9V3Eh0X-Vpsr?usp=sharing`).

2 Background

2.1 Adversarial Robustness

2.1.1 Adversarial examples

Adversarial examples can be created in several ways and can target different threat models. Common threat models are perturbations bounded by some L^p norm equal to ε, or perturbations embedded in so-called patches (Brown et al. 2017). In this work, we focus on L^p-bounded perturbations, as it is the most common threat model studied by previous work; in particular, we focus on L^∞ norm perturbations.

Adversarial examples can be either *untargeted* –when they aim at fooling a classifier into outputting any other class than the correct one– or *targeted* –when they aim at fooling a classifier into outputting a specific, wrong class. We focus this work on untargeted attacks, which are stronger than the latter. In the case of L^p-bounded perturbations, given an input $\boldsymbol{x}$ with label y, and a loss function $\mathcal{L}$, we can find an adversarial perturbation $\hat{\boldsymbol{\delta}}$ by solving the following constrained optimization problem:

$$\hat{\boldsymbol{\delta}} = \underset{\boldsymbol{\delta}:\|\boldsymbol{\delta}\|_p \leq \varepsilon}{\arg\max}\, \mathcal{L}(\boldsymbol{x} + \boldsymbol{\delta}, y). \tag{1}$$

The objective is usually optimized via *projected gradient descent* (PGD), which consists of a gradient descent step, after which we project the perturbation inside the allowed boundary (e.g., in the case of $p = \infty$, we clip each component of the perturbation to be in $[-\varepsilon, \varepsilon]$), and project the input $\boldsymbol{x} + \boldsymbol{\delta}$ to be in the allowed domain (e.g., in $[0, 1]$ in the case of images).

One variation of PGD that can be used to find adversarial examples that target the ℓ_∞ threat model is the *Fast Gradient Sign Method* (FGSM) (I. J. Goodfellow et al. 2014). FGSM, instead of adding the gradient at each step of PGD, does just one step, and adds the component-wise sign of the gradient multiplied by ε, instead of the gradient itself. Formally, given a set of allowed inputs $\mathbb{S}$ and a projection operator *Proj*, we can find an adversarial example $\hat{\boldsymbol{x}}$ from an input $\boldsymbol{x}$ with FGSM using the following formula:

$$\hat{\boldsymbol{x}} = \mathrm{Proj}_{\mathbb{S}}\left(\boldsymbol{x} + \varepsilon \mathrm{sign}(\nabla_{\boldsymbol{x}} \mathcal{L}(\boldsymbol{x}, y))\right). \tag{2}$$

To generate adversarial examples with PGD or FGSM, we need to compute the model's gradient, w.r.t. the input $\boldsymbol{x}$. For this reason, we need white-box access to the model, i.e., we need access to the model internals and parameters. There are also black-box attacks, which only have access to

the value of the loss of the model, and optimize equation (1) via zeroth-order optimization techniques, such as finite differences (Chen et al. 2017) and random search (Andriushchenko et al. 2020). An alternative to zeroth-order optimization is given by *transfer attacks*, which consists of generating an adversarial example for a model for which we have white-box access. Then, the adversarial example will likely transfer successfully to the victim model to which we have black-box access (Papernot et al. 2017).

2.1.2 Adversarial training

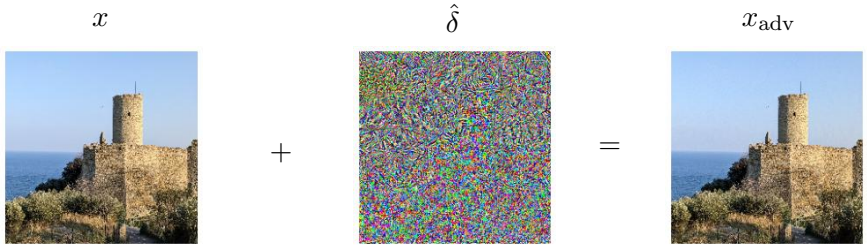
There are several methods to defend from adversarial examples. One line of work focuses on training models in a way that makes them resistant to adversarial examples. The most studied and developed technique is *adversarial training* (Madry et al. 2017). Adversarial training consists, at each step, in generating adversarial examples 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:

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}} \left[\max_{\boldsymbol{\delta}\in\mathbb{S}} \mathcal{L}(\mathbf{x}+\boldsymbol{\delta}, \mathrm{y}; \boldsymbol{\theta}) \right]. \tag{3}$$

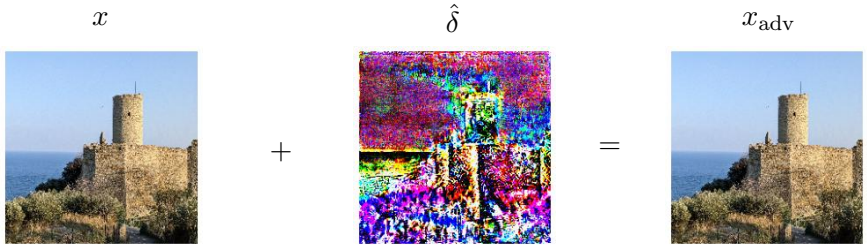
We can note that the inner optimization problem consists of generating an adversarial example for $\boldsymbol{x}$, while the outer one consists of training the model. In practice, we solve the outer optimization with stochastic gradient descent or one of its variants and solve the inner one as explained in section 2.1.1. It is also possible to use 1-step FGSM for the inner optimization to save training time (Wong et al. 2020). While adversarial training is successful in improving robustness to adversarial examples, on the other hand, it harms the accuracy of the model when classifying clean data (Tsipras et al. 2018).

Several further techniques have been proposed to improve adversarial training and the trade-off. Some of them include the optimization of a different objective (Hongyang Zhang et al. 2019) to improve the clean and robust accuracy trade-off, early stopping to avoid the so-called robust overfitting (Rice et al. 2020), tuned data augmentation, weight averaging (Rebuffi et al. 2021) etc.

TRADES. Hongyang Zhang et al. (2019) propose a different formulation for adversarial training, with the aim of improving the accuracy-robustness trade-off. This formulation, called TRadeoff-inspired Adversarial DEfense



(a) Example of how an adversarial example can be created for a non-robust model (ResNet-50 from Wightman (2019)): an image correctly classified as "castle" that is misclassified as "monastery" when we add the perturbation.



(b) Example of robust model (ResNet-50 from Bai et al. (2021)) trained with adversarial training: an image correctly classified as "castle" is still correctly classified when we add the perturbation. We can also note how the perturbation generated for the robust model is different in nature from the one for the non-robust one.

Figure 1: Differences between a non-adversarially and an adversarially trained model, given perturbations found using 10 steps PGD with $\varepsilon = 4/255$.

via Surrogate-loss minimization (TRADES), add a regularization term to the conventional loss used for standard training, which is meant to give robustness to the model being trained. In particular, given the function of the model $f(\boldsymbol{x}; \boldsymbol{\theta})$ which takes as input $\boldsymbol{x}$ and is parametrized by $\boldsymbol{\theta}$, the TRADES objective consists of solving the following optimization problem

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{(\mathbf{x},\mathrm{y})\sim p_{\text{data}}} \left[\mathcal{L}(\mathbf{x}, \mathrm{y}; \boldsymbol{\theta}) + \beta \max_{\boldsymbol{\delta}\in\mathbb{S}} D_{\text{KL}}(f(\boldsymbol{x}; \boldsymbol{\theta}) \| f(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta})) \right], \tag{4}$$

where β is a coefficient chosen to weight the importance of the second term, and $D_{\text{KL}}(f(\boldsymbol{x}; \boldsymbol{\theta}) \| f(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta}))$ is the Kullback-Leibler divergence (KL divergence) of $f(\boldsymbol{x}; \boldsymbol{\theta})$ (the output of the model given clean inputs) and

$f(\boldsymbol{x} + \boldsymbol{\delta}; \boldsymbol{\theta})$ (the output of the model given adversarially perturbed inputs). The KL divergence of two distributions P and Q measures the difference between these distributions and is defined as

$$D_{\mathrm{KL}}(P\|Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right), \tag{5}$$

where $p(x)$ and $q(x)$ are the probability density functions of P and Q respectively. In equation (4), the KL divergence measures how much the output of the model given the clean inputs differs from the output given the adversarially perturbed inputs. We can also note that, in equation (4), there are two terms: one is the loss of the model on the clean data, and the other term, weighted by a coefficient β, is the KL divergence given the adversarial examples. This coefficient can be tuned to choose a balance for the accuracy-robustness trade-off, i.e., a larger β will give better robustness, but worse clean accuracy, and vice-versa for a smaller β.

2.1.3 Evaluation of defenses

Of course, the defenses proposed in the literature must be evaluated reliably and effectively. On one side, it is possible to use standard attacks, such as projected gradient descent (PGD). Using a larger number of steps helps in increasing the reliability of the evaluation. However, some defense techniques can result in gradient masking (Papernot et al. 2017), which can make the PGD objective too hard to optimize due to the degradation of the model's gradients and give a false sense of security(Athalye et al. 2018). For this reason, we can either use other standard attacks which find adversarial examples by doing zeroth-order optimization, or we can design an adaptive attack (Tramèr et al. 2020). Using an adaptive attack can ensure that the evaluation is accurate, but designing one can take time, and it is not something that can be easily standardized. For this reason, currently, the most established evaluation technique is AutoAttack (Croce et al. 2020c), an ensemble of four different parameter-free white- and black-box attacks. In particular, the attacks are APGD-CE (Croce et al. 2020c), APGD-T (Croce et al. 2020c), FAB (Croce et al. 2020b), and Square Attack (Andriushchenko et al. 2020).

2.1.4 Adversarial robustness and interpretability

Gradients of non-robust models usually have no semantic meaning, and they do not enable visualizing the model's features in a meaningful way unless

we use strong regularization (Olah et al. 2017). On the other side, previous work by Engstrom et al. (2019) shows that gradients of models trained robustly with adversarial training are more aligned with human perception. This enables several new interesting visualizations, such as *direct feature visualization* (Engstrom et al. 2019). To produce this visualization, Engstrom et al. (2019) choose one activation in the penultimate layer of the robust model and maximize it by optimizing the input via PGD, using a large ε as a constraint for the L^p norm. By starting from images initialized randomly or from images from a dataset, they observe that the results show features that make sense to the human eye, suggesting that gradients of robust models are indeed more aligned with human perception.

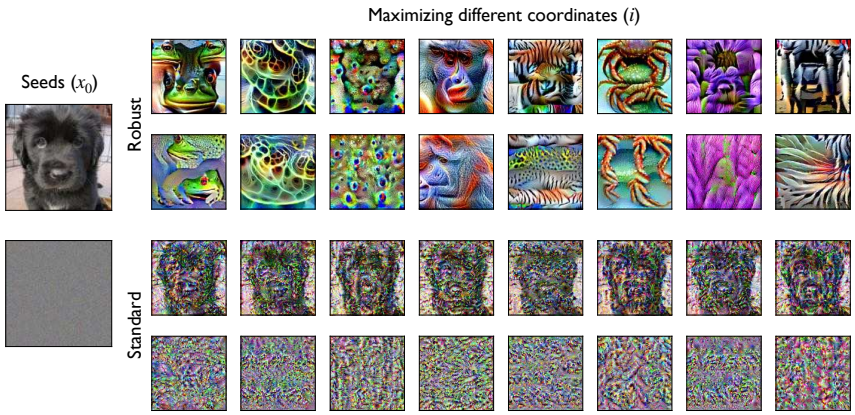


Figure 2: Examples of direct feature visualization for random- and non-random inputs, to maximize different activations. Borrowed from Engstrom et al. (2019).

2.2 Convolutional Neural Networks and ResNet

Convolutional Neural Networks (CNNs) (LeCun et al. 1989) are Neural Networks introduced at the end of the 80s that make use of *convolutional layers*. Instead of computing the dot product between the input and a learnable weight matrix, as in fully connected layers, convolutional layers compute a convolution between the input and a learnable kernel. These layers can be used for inputs with a grid-like topology both in 1-D (e.g., time-series), 2-D (e.g., gray-scale images), and 3-D (e.g., RGB images) (I. Goodfellow et al. 2016). Using convolutions brings several advantages that include a smaller memory footprint (given that the convolutional kernel usually has

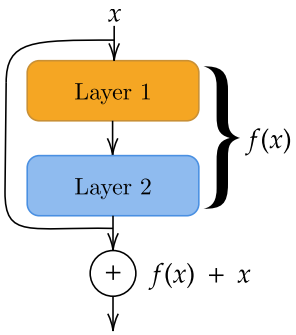


Figure 3: Illustration of a residual connection. Figure reproduced from He et al. (2015).

a small size) and a great fit to detect local features, which is enabled by the very nature of the convolution operator. Moreover, convolutions are translation invariant, meaning that they can detect specific features in an image independently of the location of such features. For these reasons, they have been the *de facto* standard for computer vision tasks since the work by Krizhevsky et al. (2012). In particular, they train a deep CNN by leveraging an efficient GPU implementation.

A further breakthrough is introduced with *ResNet* (He et al. 2015), an architecture including an innovation which enables training deeper networks to improve the performance. This improvement consists of *residual (or shortcut) connections*. If we consider a series of operations $f(x)$, a residual connection consists in taking as result $f(x) + x$, instead of just $f(x)$. This simple idea is illustrated in figure 3. Thanks to residual connections, the optimization procedure converges to better results.

2.3 Vision Transformers and their variants

2.3.1 Attention and the Transformer architecture

The Transformer architecture (Vaswani et al. 2017) is an architecture for sequence transduction and Natural Language Processing (NLP) tasks (e.g., machine translation) based on attention mechanisms. This architecture has an encoder-decoder structure, where both the encoder and the decoder include a series of so-called *Multi-Head Attention* layers, each followed by a *Multi-Layer Perceptron* layer. Each 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}.

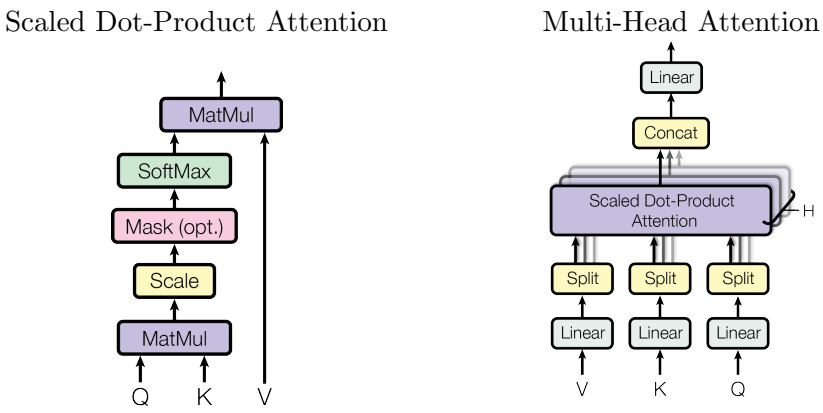


Figure 4: Schematic drawings representing the various steps of scaled dot-product attention and multi-head attention. Figure borrowed from Vaswani et al. (2017).

Attention. An 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. (2017), called *Scaled Dot-Product Attention*, can be formally expressed as

$$\text{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}, \tag{6}$$

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 Edoardo's thesis", "this" will attend to "is", which will, in turn, attend to "thesis". 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. This is a useful property in NLP because a word at the beginning of a sentence could be strongly related to another word at the end of it.

Multi-Head Attention. Before passing $\boldsymbol{Q}$, $\boldsymbol{K}$, and $\boldsymbol{V}$ to the Attention function, Vaswani et al. (2017) 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,

$$\begin{aligned} \text{MultiHead}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) &= \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\boldsymbol{W}^O, \\ \text{where head}_i &= \text{Attention}(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V), \end{aligned} \tag{7}$$

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 $\boldsymbol{K} = \boldsymbol{V}$.

The MLP block and the overall Transformer block. After computing self-attention for the inputs, Vaswani et al. (2017) pass the result 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

$$\text{MLP}(\boldsymbol{x}) = \rho(\boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2. \tag{8}$$

Vaswani et al. (2017) use the Rectified Linear Unit (ReLU) as activation function. Moreover, they apply layer normalization (Ba et al. 2016) both after the MSA and the MLP residual connections.

To summarize, given an input x_l to the l-th 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:

$$\begin{aligned} \boldsymbol{x}'_l &= \text{LN}(\boldsymbol{x}_l + \text{MSA}(\boldsymbol{x}_l)) \\ \boldsymbol{x}_{l+1} &= \text{LN}(\boldsymbol{x}'_l + \text{MLP}(\boldsymbol{x}'_l)). \end{aligned} \tag{9}$$

Positional encoding. We note 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. (2017) add the

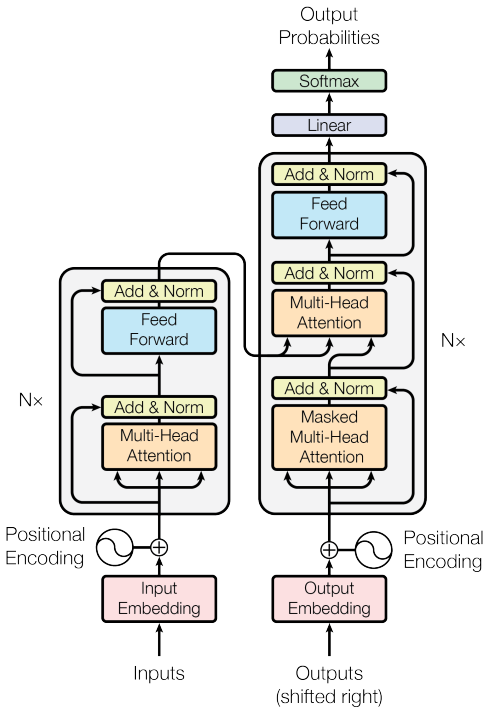


Figure 5: Schematic figure representing the overall structure of the original Transformer architecture. Figure borrowed form Vaswani et al. (2017)

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}.They adopt sine and cosine functions of different frequencies, which are fixed (i.e., not learned):

$$\begin{aligned} \text{PE}_{pos,2i} &= \sin(pos/10000^{2i/d_{\text{model}}}) \\ \text{PE}_{pos,2i+1} &= \cos(pos/10000^{2i/d_{\text{model}}}), \end{aligned} \tag{10}$$

where i is the dimension and pos is the position.

2.3.2 Vision Transformers

Overall structure. The Transformer architecture can also be easily adapted for computer vision tasks (Dosovitskiy et al. 2021). The resulting architecture is called *Vision Transformer* (ViT). In particular, it is possible to divide the input images into non-overlapping patches, which are then embedded into tokens. The biggest advantage of using attention for computer vision

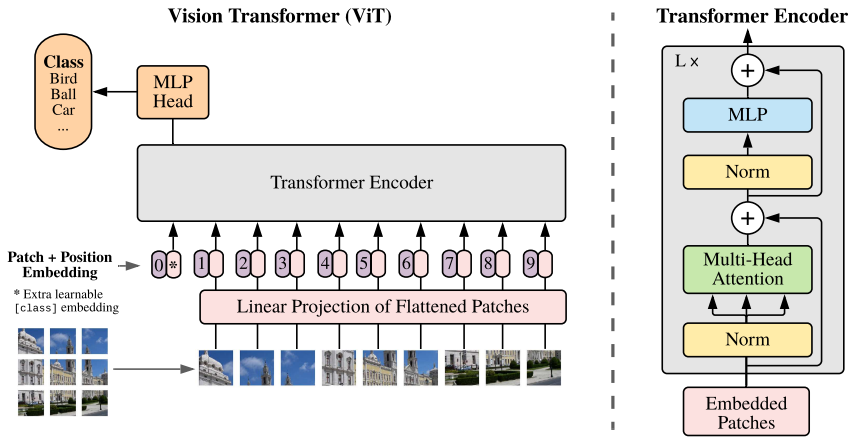


Figure 6: Schematic figure representing the overall structure of the original Vision Transformer architecture. Figure borrowed form Dosovitskiy et al. (2021).

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. Given the nature of image classification, there is no need for an encoder-decoder structure: Dosovitskiy et al. (2021) simply use the same structure as the encoder, i.e., input embedding, followed by a series of blocks composed of an MSA block and an MLP block. Moreover, at the end of the last block, there is the so-called "MLP Head": an MLP which takes as input a special token, called `[class]` token (discussed below), resulting from the last attention block, and maps it to the class predicted for the input. Moreover, differently from the original transformer architecture, they use Gaussian Error Linear Unit (GELU) (Hendrycks et al. 2016) as activation function ρ in the MLP block (equation (8)), and they apply layer normalization to the input of each MSA and MLP block, instead of the output of the respective residual connections. To summarize, equation (9) becomes:

$$\begin{aligned} \boldsymbol{x}'_l &= \boldsymbol{x}_l + \mathrm{MSA}(\mathrm{LN}(\boldsymbol{x}_l)) \\ \boldsymbol{x}_{l+1} &= \boldsymbol{x}'_l + \mathrm{MLP}_{\mathrm{GELU}}(\mathrm{LN}(\boldsymbol{x}'_l)). \end{aligned} \tag{11}$$

Input tokenization and positional encoding. Considering each pixel as a token and computing attention between every pixel would be unfeasible, as the attention operation has $\mathcal{O}(n^2)$ complexity for both memory and

runtime. For this reason, Dosovitskiy et al. (2021) 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 $\mathbb{R}^d_{\text{model}}$. In practice, the projection is efficiently implemented with a convolution layer that has as stride the patch size, as input channels the number of channels of the input image (e.g., 3 in case of RGB images), and as output channels d_{model}. Given an image of size (H, W) and patches of size (P, P), the resulting number of patches is $N = HW/P^2$.

A further difference between the architecture proposed by Dosovitskiy et al. (2021) and the original Transformer (Vaswani et al. 2017) is the fact that the positional encodings, instead of being fixed and pre-determined, are learnable.

Class token. After the input tokens are generated, a vector is prepended to the sequence of tokens and processed along with the other tokens. This vector, called `[class]` token, is then taken from the results of the last attention block and is passed to the classification head that computes the class predicted for the input. The initial state of the vector (i.e., the vector that is prepended to the tokens before being processed) 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.

Performance of Vision Transformers. Vision transformers achieve state-of-the-art performance on several datasets, such as ImageNet (Deng et al. 2009), CIFAR-10, and CIFAR-100 (Krizhevsky 2009). In particular, their maximum potential is reached when they are pre-trained on larger datasets, such as ImageNet-21k and JFT-300M (Sun et al. 2017). 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 by Touvron et al. (2021a) and Steiner et al. (2021) shows that a tuned training recipe, strong regularization, and data augmentations, such as CutMix (Yun et al. 2019), RandAugment (Cubuk et al. 2020), MixUp (Hongyi Zhang et al. 2017), and RandomErasing (Zhong et al. 2020), lead to a ten-fold decrease in the need of data to achieve the same performance. In particular, Touvron et al. (2021a) call the model trained with this training recipe *DeiT*.

2.3.3 Class Attention in Transformers

In order to train deeper vision transformers, Touvron et al. (2021b) introduce two innovations: *LayerScale* and *Class Attention*.

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 $\text{diag}(\lambda'_{l,1}, \dots \lambda'_{l,d})$, equation (11) becomes

$$\begin{aligned} \boldsymbol{x}'_l &= \boldsymbol{x}_l + \text{diag}(\lambda_{l,1}, \dots \lambda_{l,d})\,\text{MSA}(\text{LN}(\boldsymbol{x}_l)) \\ \boldsymbol{x}_{l+1} &= \boldsymbol{x}'_l + \text{diag}(\lambda'_{l,1}, \dots \lambda'_{l,d})\,\text{MLP}(\text{LN}(\boldsymbol{x}'_l)). \end{aligned} \tag{12}$$

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.

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. (2021b) first process the input tokens through a series of Transformer blocks (according to equation (12)), 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:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{W}_q \boldsymbol{x}_{\text{class}} + \boldsymbol{b}_q \\ \boldsymbol{K} &= \boldsymbol{W}_k \boldsymbol{z} + \boldsymbol{b}_k \\ \boldsymbol{V} &= \boldsymbol{W}_v \boldsymbol{z} + \boldsymbol{b}_v. \end{aligned} \tag{13}$$

They then compute class attention as

$$\text{CA}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{W}_o \text{Softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d/h}}\right)\boldsymbol{V} + \boldsymbol{b}_o, \tag{14}$$

where $\boldsymbol{QK}^T \in \mathbb{R}^{h \times 1 \times p}$, 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).

2.3.4 Cross-Covariance Attention and XCiT

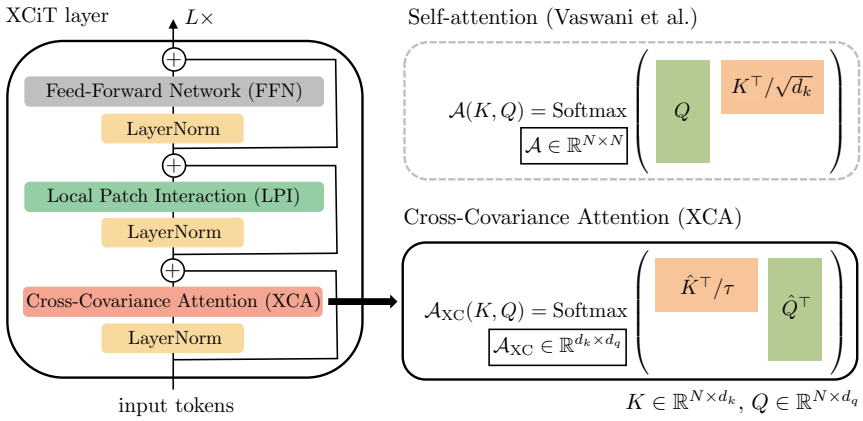


Figure 7: Schematic figure representing the overall structure of the XCiT block. Figure borrowed from El-Nouby et al. (2021).

Dosovitskiy et al. (2021) show that using a smaller patch size brings better results. However, as already mentioned in section 2.3.2, 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. (2021) 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 section 2.3.3).

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 $\boldsymbol{V}$, as

defined in section 2.3.1, cross-covariance attention is defined as

$$\text{XC-Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \boldsymbol{V}\text{Softmax}\left(\frac{\hat{\boldsymbol{K}}^T\hat{\boldsymbol{Q}}}{\tau}\right), \tag{15}$$

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.

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. (2021) 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.

Convolutional Patch Projection. Differently than the previous works about ViTs introduced above, following Graham et al. (2021), El-Nouby et al. (2021) 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. (2017).

2.3.5 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. (2021a) are *MixUp*, *CutMix*, *RandAugment*, and *Random Erasing*.

MixUp and CutMix. MixUp (Hongyi Zhang et al. 2017) 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 (Yun et al. 2019) 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 figure 8.

(a) MixUp example. (b) CutMix example.

Figure 8: Examples for MixUp *(Left)* and CutMix *(Right)* data augmentations.

RandAugment. RandAugment (Cubuk et al. 2020) 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 figure 9.

Random Erasing. Finally, Random Erasing (Zhong et al. 2020) randomly selects a portion of pixels in an image, and occludes them, either by setting

Figure 9: Original image *(Left)* and three examples of augmented images generated by RandAugment

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 figure 10.

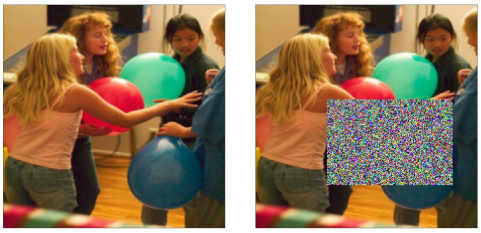


Figure 10: Original image *(Left)* and example of erasure generated by Random Erasing.

3 Related work

3.1 Improving the robustness-accuracy-efficiency trade-off

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. (2019) proposes to re-use adversarial examples from previous epochs to avoid generating new ones every time, while Wong et al. (2020) find an effective way to perform adversarial training with just 1-step FGSM. On the other hand, Sehwag et al. (2020) and Kundu et al. (2020) 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 (Carmon et al. 2019), or synthetic data (Sehwag et al. 2021; Gowal et al. 2021) when performing adversarial training.

3.2 Robustness of Vision Transformers

3.3 Robustness against 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 one side, benignly trained ViTs show to be more robust than CNNs to non-adversarial perturbations, while on the other side, different works reported contrasting results when it came to adversarial perturbations, depending on the strength of the attack.

3.3.1 Natural perturbations

Many recent works (Paul et al. 2021; Bhojanapalli et al. 2021; Bai et al. 2021) agree on the fact that ViTs are more robust than CNNs when it comes to out-of-distribution samples (the ImageNet-R dataset by Hendrycks et al. (2021)), as well as perturbations such as common corruptions (the ImageNet-C dataset by Hendrycks et al. (2019a)), natural adversarial examples (the ImageNet-A dataset by Hendrycks et al. (2019a)), and Stylized-ImageNet (Geirhos et al. 2018). ViTs are compared to both ResNets (Bhojanapalli et al. 2021; Bai et al. 2021) and Big Transfer (BiT,Kolesnikov et al. (2020)) models (Paul et al. 2021). In particular, Bai et al. (2021) compare ResNets to DeiT models. For a fair comparison, they tune both ResNet-50 and ResNet-101 to achieve the best performance possible on these datasets. Despite this, DeiT still shows better robustness. On the other hand, Bhojanapalli et al. (2021) pre-train

ResNets and ViTs of different sizes on both JFT-300M (Sun et al. 2017) and ImageNet-21k, two large datasets commonly used to pre-train large models. They observe that also when the models are pre-trained on large datasets of equal size, ViTs are more robust than ResNets. Moreover, they also observe that, when scaling up to larger models, ViTs give a larger advantage on natural corruptions than the advantage brought by larger ResNets.

ViTs show better robustness to patch corruptions as well. They are more robust in the case of image occlusions (by applying CutOut (DeVries et al. 2017)), as observed by Paul et al. (2021), as well as to naturally perturbed patches: Gu et al. (2021) built images in which they perturb only a limited number of selected patches using augmentations like those of ImageNet-C. ViTs are more robust in this case as well.

3.3.2 Adversarial perturbations

Weak perturbations. 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 using $\varepsilon \leq 0.01$ or weak attacks such as PGD, ViT-like models, as well as ViT-CNN hybrids (Dosovitskiy et al. 2021) show better robustness than CNNs (Bhojanapalli et al. 2021; Aldahdooh et al. 2021; Benz et al. 2021). On the other hand, Shao et al. (2021) find that adding convolutional blocks can help in improving clean accuracy but at the expense of robust accuracy. However, they also observe that increasing the ratio of attention blocks helps to achieve better robust accuracy. Hence, ViT-CNN hybrids are considered to be less robust than pure ViTs. Finally, using a larger model such as ViT-L or employing convolutional layers to embed the patches does not necessarily help robustness (Aldahdooh et al. 2021; Shao et al. 2021).

The better results obtained with attention-based models may be due to several reasons. Benz et al. (2021) observe that ViT-like models leverage low-frequency features of images, such as shapes, while adversarial perturbations are usually high-frequency. Moreover, Paul et al. (2021) observe that ViTs have a smoother loss landscape in the input space: previous work finds that smoothness of models is indeed necessary to achieve both high clean and robust accuracy (Yang et al. 2020). Benz et al. (2021) instead claim that the lower robustness of CNNs is given by their shift-invariant property: this property enables the adversary to fool a CNN with adversarial features in any place of the image, instead of a limited area of the input. Finally, Shao et al. (2021) hypothesize that ViTs are more robust because they learn more generalizable, high-level information, instead of low-level one.

Strong perturbations. When using slightly stronger attacks such as the C&W attack (Carlini et al. 2017) or DeepFool (Moosavi-Dezfooli et al. 2016) ViTs achieve very low robust accuracy ($< 1\%$) but are still slightly better than ResNets. However, when using stronger attacks, such as APGD or AutoAttack used with $\varepsilon = 4/255$, both ResNets and ViTs have 0% robust accuracy (Shao et al. 2021; Bai et al. 2021; Mahmood et al. 2021).

Transferability. In different works, it is also observed that, in general, the transferability of adversarial examples is low (Bhojanapalli et al. 2021; Shao et al. 2021; Aldahdooh et al. 2021). All these works note that, on one side, when the architecture is similar, the transferability is better than when the architecture is different. However, when the architecture is different, then the transfer rate decreases. Nonetheless, Shao et al. (2021) and Benz et al. (2021) observe that the transfer rate from ViTs to CNNs is slightly larger than the opposite.

Attacks. As previously observed, transfer attacks do not work very well between ViTs of different sizes nor between ViTs and CNNs. For this reason, Wei et al. (2021) and Naseer et al. (2021) develop new attacks to improve the transferability of adversarial examples targeting ViTs. On one hand, Wei et al. (2021) propose an attack based on two concepts: *PayNoAttention* and *PatchOut*, both aiming at reducing the model-specificity of the generated adversarial examples. The former consists of not considering the Attention blocks when computing the gradient with respect to the input, and the latter consists in optimizing only a random subset of the input patches instead of the whole input. On the other hand, Naseer et al. (2021), propose *Self Ensemble*, which consists of a model that uses an MLP Head that takes as input the `[class]` tokens output by all the transformer blocks, instead of just the last one. In this way, the attack targets the full representation capacity of the model.

3.4 Adversarial training for Vision Transformers

While many works focus on the adversarial robustness of transformers trained without adversarial training, to the best of our knowledge, only two works perform adversarial training of ViT-like models and study their robustness.

On the one hand, Shao et al. (2021) adversarially train ViT-B by performing adversarial fine-tuning on CIFAR-10 models that were previously pre-trained without adversarial training on ImageNet. They compare a ViT-B with a WRN-34-10 (Zagoruyko et al. 2016), and a ResNet-18 (He et al.

2015). In particular, they fine-tune the models using the original 32×32 resolution of CIFAR-10, with patch size 4, instead of using the up-scaled 224×224 resolution with patch size 16, commonly used when fine-tuning ViTs. They target the L^∞ norm with $\varepsilon = 8/255$. To adapt the model to work with this resolution, they down-sample the weight kernel of the convolutional token embedding layer of the original ViT: the original kernel has size 16×16, with stride 16, and the down-sampled one has size 4×4, with stride 4. As a result of adversarial fine-tuning, their ViT-B obtains slightly lower robust accuracy and slightly better clean accuracy than the WRN-34-10 they compare to, and better robust accuracy and clean accuracy than the PreActResNet-18. Moreover, the accuracies are all lower than many models of similar size in the corresponding RobustBench (Croce et al. 2020a) leaderboard. The reason for this could be the fact that these models are pre-trained on ImageNet without adversarial training; hence, the models they fine-tune are not robust: previous work (Hendrycks et al. 2019b) only shows that pre-training gives an advantage in adversarial robustness when done adversarially.

On the other hand, Bai et al. (2021) manage to adversarially train DeiT-S –a variant of ViT– to compare it to an adversarially trained ResNet-50. In particular, they notice that adversarially training DeiT-S with 1-step PGD, targeting the L^∞ threat model with $\varepsilon = 4/255$, fails when using the original training recipe (i.e., with full data augmentation, apart from RandomErasing). For this reason, they gradually increase the intensity of data augmentation in the first ten epochs. This method shows effective and leads to adversarially training DeiT-S successfully. On the other side, they observe that training a ResNet-50 using the same setup as DeiT with strong data augmentation fails. However, with the original training recipe and employing the GELU activation function, ResNet-50 achieves comparable performance to the adversarially trained DeiT-S. Moreover, they note that adversarially training DeiT-S without strong augmentation stabilizes the training process but leads to a big hit in performance both in terms of clean and robust accuracy. Finally, they do not ablate the data augmentations –which can play a fundamental role in adversarial training (Rebuffi et al. 2021)– for DeiT, nor try any other newer variant of ViT and do not attempt to adversarially fine-tune their robust models to other datasets.

4 Experiments

4.1 Training recipe

Training Vision Transformers and their variations is a non-trivial task, and the training recipe can make a significant difference in the final results (Touvron et al. 2021a; Steiner et al. 2021). We argue that it is necessary to find the best setup in terms of training hyperparameters to effectively train robust ViTs and that this setup may differ from the best one found for standard training. To find the best training setup, we ablate along the following axes: ViT architecture, warming up the epsilon of the PGD attack, data augmentation, and weight decay. We show that, by choosing the right recipe, we can significantly improve the performance of adversarial training of a ViT-like architecture and achieve state-of-the-art results on ImageNet.

4.1.1 Training setup

We run the ablations mentioned above on a subset of 100 randomly chosen classes of the ImageNet dataset (Russakovsky et al. 2015) –which we call ImageNet-100– to assess more quickly the choice of hyperparameters. We select the architecture using variants that are comparable to ResNet-50. However, after choosing the best architecture, we employ a smaller variant to reduce the time of the hyperparameters search. We finally scale to a larger variant when training on the full-size ImageNet dataset. ImageNet is a dataset of 1000 classes and about 1.2M training samples of variable high-resolution.

To show that the best setup for adversarial training (discussed in section 2.1.2) may differ from the best recipe found for standard training, we run the architecture, data augmentation, and weight decay ablations also for standard training.

Apart from the *Scaling up* section (section 4.1.9), we run all the training runs using machines with 8 TPUv3 cores. In all the runs, unless otherwise stated, we use the same setup as the one used for DeiT (Touvron et al. 2021a). 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. (2021a) (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, we train the model for 110 epochs, with a learning rate cosine decay with a final value of 5×10^{-5}, a 5-epochs warm-up from 5×10^{-6} and 10 epochs cool-down. Apart from the architecture ablation, we do not employ *repeated augmentations* (Hoffer et al. 2020) to save training time.

Repeated augmentations consists 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.

Finally, unless otherwise stated, we use FGSM for adversarial training (Wong et al. 2020), 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 always evaluate the checkpoint at the epoch where the model was performing best in terms of FGSM accuracy on the test set.

4.1.2 Evaluation setup

For the final ablation and the scaled-up models trained on ImageNet, we run AutoAttack (Croce et al. 2020c), an ensemble of white- and black-box attacks. We run the attack on the subset of 5000 ImageNet images used for the RobustBench benchmark (Croce et al. 2020a). 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 (Croce et al. 2020c). 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.

4.1.3 Implementation details

We base our implementation on the PyTorch Image Models repository (Wightman 2019)[4], which includes the `timm` library and provides a template training script. This library uses the PyTorch framework (Paszke et al. 2017). 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[5] 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

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

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

Image Models[6], which introduces the compatibility of the library with XLA and TPUs.

We adapt `timm`'s default training script to perform adversarial training: in particular, we add an abstraction relative to the loss computation; instead of computing it with a function, we compute the loss with a callable PyTorch `nn.Module` object, which returns three outputs: the loss, the outputs of the model given the clean data as input, and, as an `Optional` type, the outputs of the model given the perturbed data. In this way, we make the loss computation modular, and we enable computing the loss in different ways: for instance, it can be computed just on clean data, as done in standard training. As an alternative, it can be computed on the perturbed data, as done in adversarial training, or can be computed with a different loss that uses both clean and perturbed data (as in the case of the TRADES loss (Hongyang Zhang et al. 2019)).

Because of a bug in PyTorch XLA[7], 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 (Pang et al. 2020). Another issue we face is compilation time: as mentioned above, when using XLA, the computational graph of each training step must be compiled. Given that we run adversarial training, the graph representing the training step does not only include a forward and a backward pass through the model, together with the model's weights update; it also includes the attack steps, making the overall computational graph extremely large, which takes a significant time to compile (in the order of hours in the case of 10-steps TRADES). For this reason, after each attack step, we call the `xm.mark_step()` function[8] to force XLA to compile and execute the graph consisting of just one attack step, to keep the graph small enough.

Finally, the carbon footprint of the project, measured via Google Cloud's Carbon Footprint Console[9], from November up to February is 14 $kgCO_2$[10]. For scale, a flight from Paris to London generates around 55.7 $kgCO_2$ per person in economy class[11].

[6] `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.

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

[8] `https://pytorch.org/xla/release/1.10/#running-on-a-single-xla-device`

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

[10] March data is not yet available as of the date of the submission of this thesis, i.e., April 1, 2022

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

Table 1: Comparison of the different architectures, we report in bold the best results.

Architecture	Standard training	Adversarial training	
	Clean	*Clean*	*APGD-CE*
DeiT-S (Touvron et al. 2021a)	67.38	62.52	33.32
CaiT-S-12 (Touvron et al. 2021b)	80.28	70.20	35.84
XCiT-S12 (El-Nouby et al. 2021)	**90.44**	**85.06**	**54.80**

4.1.4 ViT architecture choice

After the introduction of Vision Transformers, several variations have been proposed, to solve some issues present in the first formulation of transformers (Dosovitskiy et al. 2021). In particular, we focus on the DeiT architecture (Touvron et al. 2021a) (presented in section 2.3.2), on Class-Attention in Image Transformers (CaiT) (Touvron et al. 2021b) (presented in section 2.3.3), and on the Cross-Covariance Image Transformer (XCiT) (El-Nouby et al. 2021) (presented in section 2.3.4). Additionally, both CaiT and XCiT have two class attention blocks. With this ablation, we aim at seeing whether the innovations of the more recent architectures help improve the fit to adversarial training for L^∞ norm adversarial perturbations. Since we do not use a larger dataset for pre-training (e.g. ImageNet-21k), our baseline will be the training setup introduced by the DeiT paper (Touvron et al. 2021a). In particular, we use DeiT-S (22.05M parameters, 4.61 GFLOPs), XCiT-S12 (26.25M parameters, 4.82 GFLOPs), and CaiT-S12 (25.61M parameters, 4.76 GFLOPs). All the models have input size 224×224, embedding size $d_{\text{model}} = 384$, and 12 attention blocks. They are trained with a patch size of 16 and 224×224 resolution. We aim at seeing which architecture works best with adversarial training, "out-of-the-box".

The results in table 1 suggest that Class Attention 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 the rest of our experiments. As mentioned above, to speed up the ablation study, from now on, we will be using the smallest variant of XCiT, XCiT-N12, with a patch size of 16.

Table 2: Comparison of the different numbers of ε warm-up epochs, we report in bold the best results.

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

4.1.5 Epsilon warm-up

Shao et al. (2021) observe that adversarially training a DeiT on ImageNet would fail using the same setup like 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 make the task easier for the first few epochs and then make it gradually harder. Given that we ablate the different data augmentations, we cannot do a data augmentation strength warm-up as Shao et al. (2021) do. An alternative is warming-up the ε that bounds the adversarial perturbations (Qin et al. 2019) by linearly increasing ε each epoch. We evaluate the impact of this technique by exploring a range of values (0, 5, 10, 20, 30).

We can observe in table 2 that using 20 epochs as warm-up duration gives a significant increase both in clean accuracy (+7.64%) and APGD-CE accuracy (+4.4%).

4.1.6 Data augmentation

Data augmentation plays a crucial role in adversarial training (Rebuffi et al. 2021). Moreover, Touvron et al. (2021a) and Steiner et al. (2021) remarked the importance of appropriate data augmentation and regularization for training ViTs. For this reason, we run a thorough ablation for the data augmentations used by default for training DeiTs: CutMix (Yun et al. 2019), RandAugment (Cubuk et al. 2020), MixUp (Hongyi Zhang et al. 2017), and Random Erasing (Zhong et al. 2020) (all covered in section 2.3.5). We try all 16 combinations of these augmentations while always keeping basic augmentations such as horizontal flipping, random resize-rescale, and color

Table 3: Comparison among different data augmentation strategies, we report in bold the best results.

Data Augmentation Policy				Standard training	Adversarial training	
MixUp	CutMix	RandAugment	Random Erasing	*Clean*	*Clean*	*APGD-CE*
✗	✗	✗	✓	77.22	**67.28**	**39.22**
✗	✗	✗	✗	76.60	66.78	**39.22**
✓	✗	✗	✗	76.34	61.04	38.56
✓	✗	✗	✓	76.02	60.46	38.26
✓	✓	✗	✗	76.48	62.04	38.18
✗	✗	✓	✗	**78.62**	65.34	37.64
✗	✗	✓	✓	78.08	64.76	37.62
✓	✓	✓	✓	75.32	56.64	35.38

jitter. We show the top seven set-ups in table 3, ranked by APGD-CE accuracy. We also show the results for the original setup in the last row. 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 set-ups improve APGD-CE accuracy by 3.84%. This phenomenon could be because adversarial training is a powerful enough regularizer on its own. We note that the setup with only Random Erasing has the same robust accuracy as the one with no complex augmentations but has a slightly larger clean accuracy. Despite this, we choose as the setup for the next experiments the one without Random Erasing, to keep the overall setup as simple as possible.

Moreover, we highlight that the best combination of standard training is not the same as the best one for adversarial training. If we used the same set of augmentations as standard training, we would have had a drop of 1.58% in terms of APGD-CE accuracy, and a drop of 1.94% in terms of clean accuracy. This means that the best training recipe for standard training could lead to sub-par results if used for adversarial training.

4.1.7 Weight decay

As pointed out by Pang et al. (2020), weight decay has an important role to make models more robust: as a matter of fact, a larger weight decay helps reducing the generalization gap for robust accuracy. For this reason, we ablate trying several values for the weight decay, in different orders of

Table 4: Comparison among different weight decays, we report in bold the best results.

Weight Decay	Standard training	Adversarial training	
	Clean	*Clean*	*APGD-CE*
0	76.32	66.44	39.02
0.001	76.32	66.40	39.04
0.01	76.26	66.28	38.66
0.05	76.36	67.16	39.30
0.1	76.58	67.28	39.92
0.5	**76.72**	**68.78**	**42.02**
1.0	76.12	67.68	40.88

magnitude: 0, 0.001, 0.01, 0.05, 0.1, 0.5, 1. We show the results in table 4. The weight decay used to train XCiT originally is 0.05, but we get the best results with the weight decay equal to 0.5, both in terms of clean and robust accuracy, with a boost of 2.72% for robust accuracy and 1.62% for clean accuracy over the original weight decay (0.05). We also observe that, in this case, weight decay does not make a big difference for standard training. Hence, basing the adversarial training weight decay decision on the standard training results could be suboptimal.

4.1.8 Final ablation and discussion

Ablation setup. As already mentioned in section 4.1.1, we finally run a step-by-step ablation on a larger scale by employing larger models trained on the full-scale ImageNet dataset. We validate these results using AutoAttack.

In particular, starting from the original training recipe used for standardly trained DeiT-S, we evaluate 1) the impact of the addition of Class Attention and Cross-Covariance attention, 2) the impact of using a warm-up for ε 3) the impact of better-tuned data augmentation, and 4) the impact of better-tuned weight decay. For the warm-up we use a schedule of 10 epochs, instead of 20, which was the warm-up duration that led to the best results on ImageNet-100. We do so as ImageNet is larger than ImageNet-100, and we argue that, after 10 epochs, the model has already seen enough *easy* samples.

Discussion. We can see the results of this ablation in table 5. We first observe that the use of XCiT leads to a greatly improved clean accuracy

Table 5: Summary of the improvements we provide. Overall, we improve the clean accuracy by 6.04%, and the robust one by 9.08% over the baseline.

Feature	Clean accuracy	AutoAttack accuracy
DeiT-S	66.30	32.70
+ Cross-Covariance Attention (XCiT-S12)	71.68 (+5.38)	28.70 (−4.00)
+ Epsilon 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)

(+5.38%), which comes at the expense of robust accuracy. However, this gap is filled with successive improvements. In particular, the improvement given by the tuned data augmentation, as well as the one given by the tuned weight decay, brings a significant boost in terms of robust accuracy. This improvement totals up to an improvement of 12.42%. Overall, the final model has 72.34% clean accuracy, and 41.78% robust accuracy. These results are better than the top entry of the RobustBench ImageNet L^∞ leaderboard: a WideResNet-50-2 (69M parameters, 11.47 GFLOPs) from Salman et al. (2020) which has 68.46% clean accuracy and 38.14% robust accuracy. Moreover, we improve over the results from Bai et al. (2021): their GELU ResNet-50, which was thoroughly ablated, achieves 67.38% clean accuracy and 35.51% robust accuracy.

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. (2021) and Touvron et al. (2021a). On the other side, we note that we use *minimal* data augmentation on the smaller ImageNet-100. Hence, we argue that, as ImageNet is larger than ImageNet-100, it should require the same (if not even weaker) data augmentation. 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, we can see in our ablation that using a larger weight decay gives a significant 3% robust accuracy improvement, so this may not be the case for adversarial training.

4.1.9 Scaling up

Setup. Given that our training configuration scaled successfully from XCiT-N12 to XCiT-S12, we check whether it scales up to even larger architectures: XCiT-M12 (which has $d_{\text{model}} = 512$, i.e., $1.5\times$ the size of XCiT-S12) and XCiT-L12 (which has $d_{\text{model}} = 768$, i.e., $2\times$ the size of XCiT-S12). Both XCiT-S12 and XCiT-M12 have 8 heads in the Multi-Head-Attention layers, while XCiT-L12 has 16. Given the larger size of these models, we do the training using a pod with 64 TPUv4 cores (as opposed to 8 TPUv3 cores). For the sake of fairness, we also train an additional XCiT-S12 model on a machine 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 section 4.1.1. The total training time for XCiT-S12 is 19h30m, for XCiT-M12 it is 33h, and for XCiT-L12 it is 39h.

Discussion. We show the results in table 6. We note that XCiT-M12 brings a significant improvement over XCiT-S12, both in terms of clean accuracy (+1.7%) and robust accuracy (+3.46%). Moreover, the performance of both XCiT-S12 and XCiT-M12 is better than that of the WideResNet-50-2, which not only has more parameters but also more GFLOPs. Finally, interestingly, XCiT-L12 has worse robust accuracy than XCiT-M12. When looking deeper into this, we note that XCiT-L12 has better FGSM accuracy than XCiT-M12 (55.88% vs. 53.82%) on the RobustBench ImageNet subset, as well as better PGD-10 accuracy (52.22% vs. 51.50%). 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%). These numbers suggest that under stronger attacks the advantage of XCiT-L12 vanishes completely. This could be because XCiT-L12 may require further tuning in the training recipe to leverage its full potential, given both the different embedding sizes, as well as the different number of heads.

To better understand the trade-off, we also standardly train an XCiT-S12, XCiT-M12, and an XCiT-L12, for 100 epochs, using the recipe from El-Nouby et al. (2021) (apart from repeated augmentations). XCiT-S12 has 80.36% clean accuracy (compared to 72.34% clean accuracy for the robust model), XCiT-M12 has 81.71% clean accuracy (compared to 74.04% clean accuracy for the robust model), and XCiT-L12 has 82.65% clean accuracy (compared to 74.60% clean accuracy for the robust model). Overall the relative drop in terms of clean accuracy is around 10% for all the models.

Finally, we compare our XCiT-M12 (46M parameters, 8.54 GFLOPs) to

the ResNeXt-152-32x8d with SiLU activation function from Xie et al. (2020) (120M parameters, 66.26 GFLOPs). They report 78.2% clean accuracy and 51.2% PGD-200 accuracy; while the clean accuracy is better than the one of XCiT-M12, our model has PGD-200 a accuracy of 51.91%, which is on-par with their result, with a model that is 2.6× smaller and has 7.75× less GFLOPs.

Table 6: Results obtained when scaling up the model size, also in comparison to robust ResNets. The GELU ResNet-50 result is from Bai et al. (2021), while the WideResNet-50-2 is from Salman et al. (2020). The robust models trained by us are in bold.

Model	d_{Model}	Parameters	GFLOPs	Clean Accuracy	AA Accuracy
GELU ResNet-50	—	25M	4.11	67.38	35.51
WideResNet-50-2	—	68M	11.47	68.46	38.14
XCiT-S12	324	26M	4.82	72.34	41.78
XCiT-M12	512	46M	8.54	74.04	45.24
XCiT-L12	768	104M	18.97	74.60	43.78

4.2 Robust fine-tuning

On the one hand, previous work showed that using additional data when doing adversarial training helps (Schmidt et al. 2018). On the other hand, ViTs give significantly better results on smaller datasets such as CIFAR-10 (Krizhevsky 2009) and VTAB datasets (Zhai et al. 2019) when they are pre-trained on larger ones (e.g. ImageNet or ImageNet-21k). Moreover, from the practical point of view, fine-tuning is extremely useful, as practitioners can fine-tune a pre-trained model on a different dataset in a smaller amount of time and compute. For this reason, we use the training setup used for ImageNet to train an $\varepsilon = {}^{8}/_{255}$-robust model and then fine-tune it on CIFAR-10, as well as other high-resolution datasets such as Caltech 101 (Fei-Fei et al. 2004) and Oxford Flowers (Nilsback et al. 2008). Caltech-101 is a dataset of pictures of resolution around 300×200 with 101 classes, very different each from the other (hence, it is coarse-grained), with about 80 to 500 images per class. Hence, the number of samples per class can be fairly small. Oxford Flowers, instead, is a dataset of 102 classes of flowers, each of which has 40 to 258 high-resolution images. Since these are all flowers, the classes do not differ very much. Hence, we consider this dataset to be fine-grained. Finally, CIFAR-10 is a dataset of images of resolution 32×32 with 10 classes and 50k

training samples.

Pre-training. As mentioned above, we pre-train a XCiT-S12 on ImageNet 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 (Herrmann et al. 2021)). 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.

4.2.1 Fine-tuning

Table 7: Summary of the results for robust fine-tuning on high-resolution datasets. The robust models trained by us are in bold. The AutoAttack accuracy of the models fine-tuned standardly is marked "—" as they are non-robust.

Fine-tuning	Model	Dataset			
		Caltech-101		Oxford Flowers	
		Clean	AutoAttack	Clean	AutoAttack
Standard	**XCiT-S12**	90.42	—	89.08	—
	ResNet-50	86.97	—	82.30	—
Adversarial	**XCiT-S12**	89.66	49.66	85.01	29.37
	ResNet-50	83.22	33.90	70.61	21.52

High resolution datasets. We fine-tune the model pre-trained on ImageNet with $\varepsilon = 8/255$ on the high-resolution datasets Caltech-101 and Oxford Flowers. We choose these datasets as the former is coarsely-grained, while the latter is fine-grained, making the set of tasks diverse. We fine-tune using $\varepsilon = 8/288$ 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. For comparison, we fine-tune, with the same set-ups for both standard and adversarial training, the ResNet-50 pre-trained

with $\varepsilon = 8/255$ from the repository of Salman et al. (2020)[12]. On the subset of 5000 ImageNet samples from RobustBench, this model has 54.90% clean accuracy and 19.72% AutoAttack accuracy.

From table 7 we can see that: 1) we can easily fine-tune both the architectures 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 drop of 0.76% w.r.t. the standardly trained model (vs. a 3.75% drop for ResNet-50), and the robust model trained on Oxford Flowers has a drop in terms of clean accuracy of 3.99% (vs. an 11.69% drop for ResNet-50). 4) Despite the larger clean accuracy, XCiT-S12 is more robust, having a robust accuracy of 13.88% better on Caltech-101, and 10.88% better on Oxford Flowers.

Adapting to small resolution images. The pre-trained model is 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 (which has 32×32 resolution). For this reason, we need a way to adapt the model to support a different patch size. Previous work (Shao et al. 2021) achieves this by down-sampling the weights of the convolutional layer that is used by ViT to embed each patch. However, as presented in section 2.3.4, XCiT uses 4 subsequent convolutional layers to embed 16×16 patches into 1-D vectors, and each layer has stride 2. 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.

CIFAR-10. Given the smaller size and resolution of the CIFAR-10 dataset, we fine-tune the robustly trained model using TRADES (Hongyang Zhang et al. 2019) (presented in section 2.1.2), 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

[12] `https://github.com/microsoft/robust-models-transfer`

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. Using this setup, we obtain a model with 90.06% clean accuracy and 56.14% AutoAttack accuracy. 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. (2021) to –at least partially– compensate for the smaller size of the dataset. By keeping the same batch size of the original setup, we set each batch to be half of real data and half of generated data. We show the results in table 8. Even though the results without pre-training are not too bad, we note that the model pre-trained on ImageNet has a significant boost in terms of both clean and robust accuracy.

Table 8: Results for CIFAR-10 fine-tuning

Training	Clean Accuracy	AA Accuracy
CIFAR-10 fine-tuning (with pre-training)	90.06	56.14
CIFAR-10 and synthetic data (from scratch)	80.01	47.88
Only CIFAR-10 (from scratch)	82.84	39.49

4.3 Semantic nature of XCiT's adversarial perturbations

Given the different nature of the architectures based on attention-like mechanisms, such as XCiT, compared to CNNs, we explore the adversarial perturbations targeted for a robust XCiT, and we compare them with those targeted for a non-robust XCiT (using the pre-trained checkpoints from the timm library (Wightman 2019)), and those targeted to the robust GELU ResNet-50 shared by Bai et al. (2021). We also explore how direct feature visualizations look like (Engstrom et al. 2019).

4.3.1 XCiT's adversarial perturbations

Perturbations visualization. We first 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.

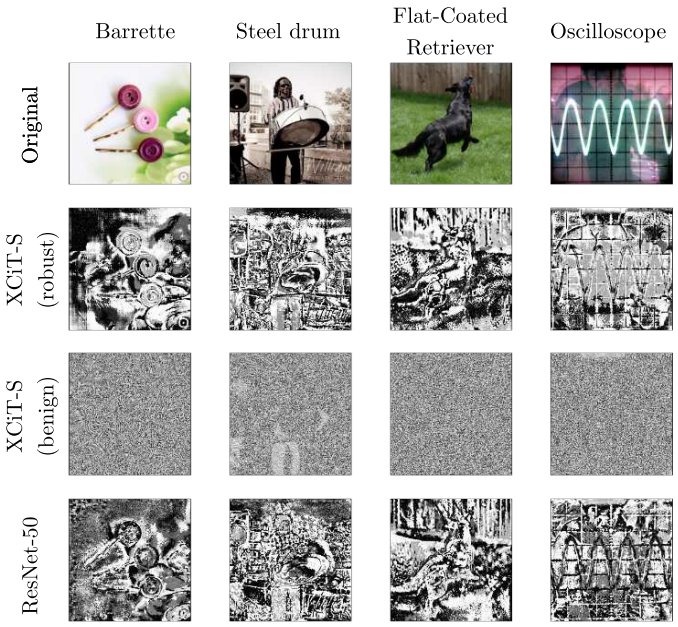


Figure 11: Comparison between the perturbations for a robust XCiT-S12 and a regular XCiT-S12.

We can see a random sample of images and their respective perturbations in figure 11 (with more images and perturbations in appendix A.1). We note that the shapes of the original images are visible in the robust XCiT perturbations, while they are not in the non-robust XCiT ones.

Quantifying the semantic nature of perturbations. To quantify how semantic the perturbations are, we propose to use high-performing models (which are trained standardly) to classify the adversarial perturbations. We argue 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 correctly by a model trained on ImageNet as the perturbations should be focused on the shapes in the input image. We use, as classifiers, a ConvNeXt-XL (Zhuang Liu et al. 2022), with 87.01% clean accuracy, a BeiT-L (Bao et al. 2022), with 87.48% clean accuracy, and a Swin-B (Ze Liu et al. 2021), with 86.32% clean accuracy, using the implementation and pre-trained weights from the timm library (Wightman 2019). 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 (Wightman 2019), as well as for a robust ResNet50 from Bai et al. (2021)[13] and a non-robust ResNet-50 from the timm library (shared by Wightman et al. (2021)). We apply to the perturbations the same scaling we apply to visualize them to bring the inputs into the $[0, 1]$ range. We can see from table 9 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 are indeed semantic, and more so than those generated for the robust ResNet-50.

Table 9: Top-5 accuracy of pre-trained ConvNeXt-XL, BeiT-L and Swin-L when classifying adversarial perturbations generated for a robust XCiT-S12 and a non-robust one.

Perturbations generator		Classifier		
		ConvNeXt-XL	BeiT-L	Swin-L
Robust	XCiT-S12	43.86	49.52	40.24
	ResNet-50	38.40	45.02	36.70
Non-robust	XCiT-S12	0.84	0.78	0.84
	ResNet-50	0.82	0.74	0.80

4.3.2 Robust XCiT's gradients

Gradients of robust CNNs are more aligned with human perception (Engstrom et al. 2019). 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 figure 12, with more images in appendix A.2.

[13] `https://github.com/ytongbai/ViTs-vs-CNNs`

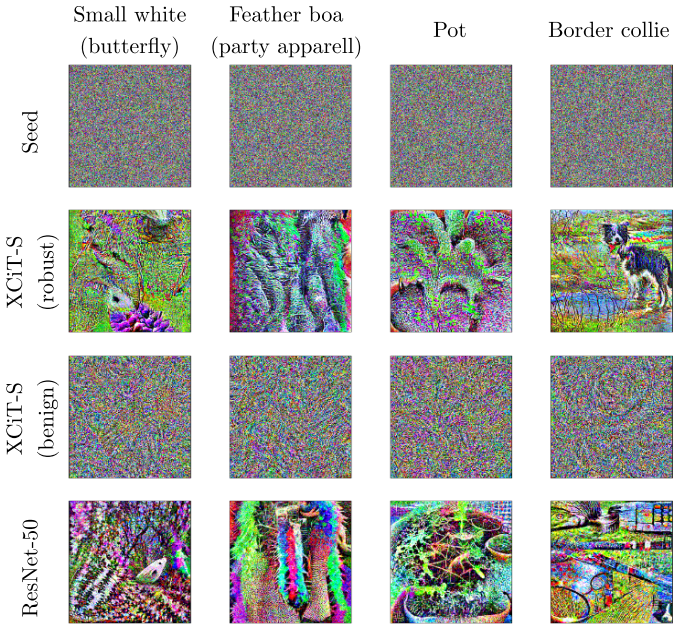


Figure 12: Comparison between the gradient accumulation for a robust XCiT-S12 and a non-robust ResNet-50.

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 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.

5 Conclusion and future work

Architecture 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 training recipe for standard training. Using this custom recipe, we achieve state-of-the-art results, by a large margin, both in terms of clean and robust accuracy. We believe that there may be other architectural innovations that could lead to even better results. Nonetheless, researchers should be aware that they may need a custom training setup to fully leverage the benefits of such architectural innovations.

Fine-tuning. We further successfully show that ViTs can be efficiently fine-tuned, for larger perturbations sizes, to very high accuracy on smaller datasets. As a matter of fact, our tailored training recipe also works for larger perturbations (i.e. $\varepsilon = 8/255$) with minimal changes. This enables efficiently fine-tuning these models to other datasets and doing adversarial training on smaller high-resolution datasets, thus improving the practicality of adversarial training. 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 ImageNet and other 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. Finally, we analyze the gradients of our robust XCiT and compare the visualizations to a state-of-the-art robust ResNet: we visualize the adversarial perturbations and 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 will release the checkpoints of our models in three variants (XCiT-S12, XCiT-M12, and XCiT-L12), and, in the case of XCiT-S12, for two perturbation sizes ($\varepsilon = 4/255$ and $\varepsilon = 8/255$). We believe that this enables researchers to do further analyses that will improve our understanding of why XCiT is particularly suitable for adversarial training.

References

Aldahdooh, Ahmed, Wassim Hamidouche, and Olivier Deforges (2021). "Reveal of vision transformers robustness against adversarial attacks". In: *arXiv preprint arXiv:2106.03734*.

Andriushchenko, Maksym, Francesco Croce, Nicolas Flammarion, and Matthias Hein (2020). "Square attack: a query-efficient black-box adversarial attack via random search". In: *European Conference on Computer Vision*. Springer, pp. 484–501.

Athalye, Anish, Nicholas Carlini, and David Wagner (July 2018). "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *Proceedings of the 35th International Conference on Machine Learning*. Ed. by Jennifer Dy and Andreas Krause. Vol. 80. Proceedings of Machine Learning Research. PMLR, pp. 274–283. URL: https://proceedings.mlr.press/v80/athalye18a.html.

Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E Hinton (2016). "Layer normalization". In: *arXiv preprint arXiv:1607.06450*.

Bai, Yutong, Jieru Mei, Alan L Yuille, and Cihang Xie (2021). "Are Transformers more robust than CNNs?" In: *Advances in Neural Information Processing Systems* 34.

Bao, Hangbo, Li Dong, Songhao Piao, and Furu Wei (2022). "BEiT: BERT Pre-Training of Image Transformers". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=p-BhZSz59o4.

Benz, Philipp, Soomin Ham, Chaoning Zhang, Adil Karjauv, and In So Kweon (2021). "Adversarial robustness comparison of vision transformer and mlp-mixer to cnns". In: *arXiv preprint arXiv:2110.02797*.

Bhojanapalli, Srinadh, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit (2021). "Understanding robustness of transformers for image classification". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10231–10241.

Biggio, Battista, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli (2013). "Evasion attacks against machine learning at test time". In: *Joint European conference on machine learning and knowledge discovery in databases*. Springer, pp. 387–402.

Brown, Tom B, Dandelion Mané, Aurko Roy, Mart ín Abadi, and Justin Gilmer (2017). "Adversarial patch". In: *arXiv preprint arXiv:1712.09665*.

Carlini, Nicholas and David Wagner (2017). "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. IEEE, pp. 39–57.

Carmon, Yair, Aditi Raghunathan, Ludwig Schmidt, John C Duchi, and Percy S Liang (2019). "Unlabeled data improves adversarial robustness". In: *Advances in Neural Information Processing Systems* 32.

Chen, Pin-Yu, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh (2017). "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models". In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*, pp. 15–26.

Croce, Francesco, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein (2020a). "Robustbench: a standardized adversarial robustness benchmark". In: *arXiv preprint arXiv:2010.09670*.

Croce, Francesco and Matthias Hein (June 2020b). "Minimally distorted Adversarial Examples with a Fast Adaptive Boundary Attack". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 2196–2205. URL: https://proceedings.mlr.press/v119/croce20a.html.

— (2020c). "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks". In: *ArXiv* abs/2003.01690.

Cubuk, Ekin D, Barret Zoph, Jonathon Shlens, and Quoc V Le (2020). "Randaugment: Practical automated data augmentation with a reduced search space". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 702–703.

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

DeVries, Terrance and Graham W Taylor (2017). "Improved regularization of convolutional neural networks with cutout". In: *arXiv preprint arXiv:1708.04552*.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *ArXiv* abs/2010.11929.

Engstrom, Logan, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry (2019). "Adversarial robustness as a prior for learned representations". In: *arXiv preprint arXiv:1906.00945*.

Fei-Fei, Li, Rob Fergus, and Pietro Perona (2004). "Learning Generative Visual Models from Few Training Examples: An Incremental Bayesian Approach Tested on 101 Object Categories". In: *Computer Vision and Pattern Recognition Workshop*.

Geirhos, Robert, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel (2018). "ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness". In: *arXiv preprint arXiv:1811.12231*.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press.

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

Gowal, Sven, Sylvestre-Alvise Rebuffi, Olivia Wiles, Florian Stimberg, Dan Andrei Calian, and Timothy A Mann (2021). "Improving robustness using generated data". In: *Advances in Neural Information Processing Systems* 34.

Graham, Benjamin, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze (2021). "LeViT: a Vision Transformer in ConvNet's Clothing for Faster Inference". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12259–12269.

Gu, Jindong, Volker Tresp, and Yao Qin (2021). "Are Vision Transformers Robust to Patch Perturbations?" In: *arXiv preprint arXiv:2111.10659*.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). "Deep residual learning for image recognition. arXiv 2015". In: *arXiv preprint arXiv:1512.03385*.

Hendrycks, Dan, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. (2021). "The many faces of robustness: A critical analysis of out-of-distribution generalization". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 8340–8349.

Hendrycks, Dan and Thomas Dietterich (2019a). "Benchmarking neural network robustness to common corruptions and perturbations". In: *arXiv preprint arXiv:1903.12261*.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415*.

Hendrycks, Dan, Kimin Lee, and Mantas Mazeika (June 2019b). "Using Pre-Training Can Improve Model Robustness and Uncertainty". In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceed-

ings of Machine Learning Research. PMLR, pp. 2712–2721. URL: https://proceedings.mlr.press/v97/hendrycks19a.html.

Herrmann, Charles, Kyle Sargent, Lu Jiang, Ramin Zabih, Huiwen Chang, Ce Liu, Dilip Krishnan, and Deqing Sun (2021). "Pyramid Adversarial Training Improves ViT Performance". In: *arXiv preprint arXiv:2111.15121*.

Hoffer, Elad, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry (2020). "Augment your batch: Improving generalization through instance repetition". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8129–8138.

Huang, Hanxun, Yisen Wang, Sarah Erfani, Quanquan Gu, James Bailey, and Xingjun Ma (2021). "Exploring Architectural Ingredients of Adversarially Robust Deep Neural Networks". In: *Advances in Neural Information Processing Systems* 34.

Kolesnikov, Alexander, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby (2020). "Big transfer (bit): General visual representation learning". In: *European conference on computer vision*. Springer, pp. 491–507.

Krizhevsky, Alex (2009). *Learning multiple layers of features from tiny images*. Tech. rep. University.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25.

Kundu, Souvik, Mahdi Nazemi, Peter A Beerel, and Massoud Pedram (2020). "A tunable robust pruning framework through dynamic network rewiring of dnns". In: *arXiv preprint arXiv:2011.03083*.

Kurakin, Alexey, Ian Goodfellow, and Samy Bengio (2016). "Adversarial examples in the physical world. arXiv". In: *arXiv preprint arXiv:1607.02533*.

LeCun, Yann et al. (1989). "Generalization and network design strategies". In: *Connectionism in perspective* 19.143-155, p. 18.

Liu, Ze, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo (Oct. 2021). "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10012–10022.

Liu, Zhuang, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie (2022). "A ConvNet for the 2020s". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu (2017). "Towards deep learning models resistant to adversarial attacks". In: *arXiv preprint arXiv:1706.06083*.

Mahmood, Kaleel, Rigel Mahmood, and Marten Van Dijk (2021). "On the robustness of vision transformers to adversarial examples". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7838–7847.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (2016). "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2574–2582.

Naseer, Muzammal, Kanchana Ranasinghe, Salman Khan, Fahad Shahbaz Khan, and Fatih Porikli (2021). "On improving adversarial transferability of vision transformers". In: *arXiv preprint arXiv:2106.04169*.

Nilsback, M-E. and A. Zisserman (Dec. 2008). "Automated Flower Classification over a Large Number of Classes". In: *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*.

El-Nouby, Alaaeldin, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, and Hervé Jégou (2021). "XCiT: Cross-Covariance Image Transformers". In: *ArXiv* abs/2106.09681.

Olah, Chris, Alexander Mordvintsev, and Ludwig Schubert (2017). "Feature Visualization". In: *Distill*. https://distill.pub/2017/feature-visualization. DOI: 10.23915/distill.00007.

Pang, Tianyu, Min Lin, Xiao Yang, Jun Zhu, and Shuicheng Yan (2022). "Robustness and Accuracy Could Be Reconcilable by (Proper) Definition". In: *arXiv preprint arXiv:2202.10103*.

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

Papernot, Nicolas, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami (2017). "Practical black-box attacks against machine learning". In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pp. 506–519.

Paszke, Adam, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer (2017). "Automatic differentiation in pytorch". In.

Paul, Sayak and Pin-Yu Chen (2021). "Vision transformers are robust learners". In: *arXiv preprint arXiv:2105.07581*.

Qin, Chongli, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli (2019). "Adversarial robustness through local linearization". In: *Advances in Neural Information Processing Systems* 32.

Rade, Rahul and Seyed-Mohsen Moosavi-Dezfooli (2021). "Helper-based Adversarial Training: Reducing Excessive Margin to Achieve a Better Accuracy vs. Robustness Trade-off". In: *ICML 2021 Workshop on Adversarial Machine Learning*. URL: https://openreview.net/forum?id=BuD2LmNaU3a.

Rebuffi, Sylvestre-Alvise, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann (2021). "Data Augmentation Can Improve Robustness". In: *Advances in Neural Information Processing Systems* 34.

Rice, Leslie, Eric Wong, and Zico Kolter (2020). "Overfitting in adversarially robust deep learning". In: *International Conference on Machine Learning*. PMLR, pp. 8093–8104.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei (2015). "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* 115, pp. 211–252.

Salman, Hadi, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry (2020). "Do adversarially robust imagenet models transfer better?" In: *Advances in Neural Information Processing Systems* 33, pp. 3533–3545.

Schmidt, Ludwig, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry (2018). "Adversarially robust generalization requires more data". In: *Advances in neural information processing systems* 31.

Sehwag, Vikash, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal (2021). "Robust Learning Meets Generative Models: Can Proxy Distributions Improve Adversarial Robustness?" In: *arXiv preprint arXiv:2104.09425*.

Sehwag, Vikash, Shiqi Wang, Prateek Mittal, and Suman Jana (2020). "Hydra: Pruning adversarially robust neural networks". In: *Advances in Neural Information Processing Systems* 33, pp. 19655–19666.

Shafahi, Ali, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein (2019). "Adversarial training for free!" In: *Advances in Neural Information Processing Systems* 32.

Shao, Rulin, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh (2021). "On the Adversarial Robustness of Vision Transformers". In: *arXiv preprint arXiv:2103.15670*.

Steiner, Andreas, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer (2021). "How to train your vit?

data, augmentation, and regularization in vision transformers". In: *arXiv preprint arXiv:2106.10270*.

Sun, Chen, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta (2017). "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*, pp. 843–852.

Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus (2013). "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199*.

Touvron, Hugo, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou (2021a). "Training data-efficient image transformers & distillation through attention". In: *International Conference on Machine Learning*. PMLR, pp. 10347–10357.

Touvron, Hugo, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Herv'e J'egou (2021b). "Going deeper with Image Transformers". In: *ArXiv* abs/2103.17239.

Tramèr, Florian, Nicholas Carlini, Wieland Brendel, and Aleksander Madry (2020). "On Adaptive Attacks to Adversarial Example Defenses". In: *Conference on Neural Information Processing Systems (NeurIPS)*. URL: https://arxiv.org/abs/2002.08347.

Tsipras, Dimitris, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry (2018). "Robustness may be at odds with accuracy". In: *arXiv preprint arXiv:1805.12152*.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need". In: *Advances in neural information processing systems* 30.

Wei, Zhipeng, Jingjing Chen, Micah Goldblum, Zuxuan Wu, Tom Goldstein, and Yu-Gang Jiang (2021). "Towards transferable adversarial attacks on vision transformers". In: *arXiv preprint arXiv:2109.04176*.

Wightman, Ross (2019). *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. DOI: 10.5281/zenodo.4414861.

Wightman, Ross, Hugo Touvron, and Hervé Jégou (2021). "Resnet strikes back: An improved training procedure in timm". In: *arXiv preprint arXiv:2110.00476*.

Wong, Eric, Leslie Rice, and J Zico Kolter (2020). "Fast is better than free: Revisiting adversarial training". In: *arXiv preprint arXiv:2001.03994*.

Xie, Cihang, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le (2020). "Smooth adversarial training". In: *arXiv preprint arXiv:2006.14536*.

Yang, Yao-Yuan, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri (2020). "A closer look at accuracy vs. robustness". In: *Advances in neural information processing systems* 33, pp. 8588–8601.

Yun, Sangdoo, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo (2019). "Cutmix: Regularization strategy to train strong classifiers with localizable features". In: *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032.

Zagoruyko, Sergey and Nikos Komodakis (2016). "Wide residual networks". In: *arXiv preprint arXiv:1605.07146*.

Zhai, Xiaohua, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. (2019). "A large-scale study of representation learning with the visual task adaptation benchmark". In: *arXiv preprint arXiv:1910.04867*.

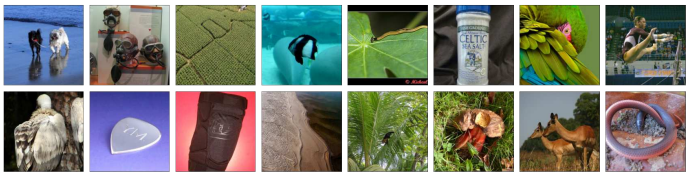
Zhang, Hongyang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan (2019). "Theoretically principled trade-off between robustness and accuracy". In: *International conference on machine learning*. PMLR, pp. 7472–7482.

Zhang, Hongyi, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz (2017). "mixup: Beyond empirical risk minimization". In: *arXiv preprint arXiv:1710.09412*.

Zhong, Zhun, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang (2020). "Random erasing data augmentation". In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 34, pp. 13001–13008.

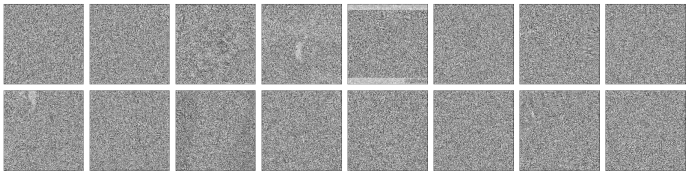
A Additional images

A.1 Adversarial perturbations visualization

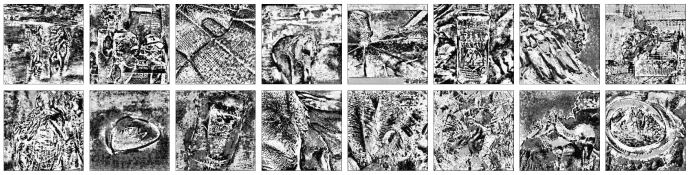


(a) Original images

(b) Robust XCiT perturbations



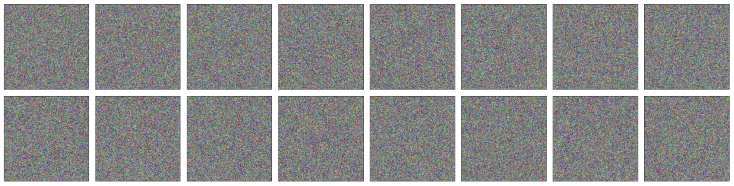
(c) Non-robust XCiT perturbations



(d) ResNet-50 perturbations

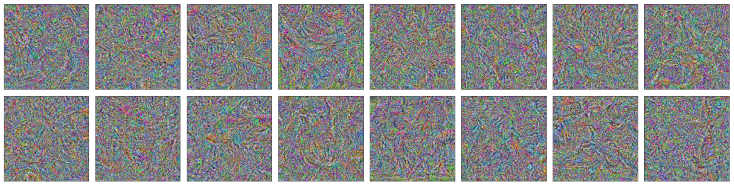
Figure 13: Visualization of the perturbations found for different images, for different models. The classes of the original images are, in order: Lhasa Apso, oxygen mask, maze, dugong, flatworm, salt shaker, macaw, horizontal bar, vulture, plectrum, knee pad, shoal, white-headed capuchin, bolete, impala, worm snake.

A.2 Gradients accumulation



(a) Seed images

(b) Robust XCiT gradients accumulation



(c) Non-robust XCiT gradients accumulation

(d) ResNet-50 gradients accumulation

Figure 14: Visualization of the gradient accumulation for different images, for different models. The target classes are, in order: Chow Chow, drum, whiskey jug, pug, purse, flamingo, lionfish, moped, barometer, T-shirt, newt, beaker, Saluki, lion, stretcher, mortar.