Towards Out-of-Distribution Adversarial Robustness

Adam Ibrahim [1 2] Charles Guille-Escuret [1 2] Ioannis Mitliagkas [1 2]
Irina Rish [1 2] David Krueger [1 2 3] Pouya Bashivan [1 4]

Abstract

Adversarial robustness continues to be a major challenge for deep learning. A core issue is that robustness to one type of attack often fails to transfer to other attacks. While prior work establishes a theoretical trade-off in robustness against different L_p norms, we show that there is potential for improvement against many commonly used attacks by adopting a domain generalisation approach. Concretely, we treat each type of attack as a domain, and apply the Risk Extrapolation method (REx), which promotes similar levels of robustness against all training attacks. Compared to existing methods, we obtain similar or superior worst-case adversarial robustness on attacks seen during training. Moreover, we achieve superior performance on families or tunings of attacks only encountered at test time. On ensembles of attacks, our approach improves the accuracy from 3.4% with the best existing baseline to 25.9% on MNIST, and from 16.9% to 23.5% on CIFAR10.

1. Introduction

While much work has been done on defending against adversarial attacks, new attacks commonly overcome existing defenses (Athalye et al., 2018). A defense that has so far passed the test of time against individual attacks is adversarial training. Goodfellow et al. (2015) originally proposed training on examples perturbed with the Fast Gradient Sign Method (FGSM), which performs a step of sign gradient ascent on a sample x to increase the chances of the model misclassifying it. Madry et al. (2018) further improved robustness by training on Projected Gradient Descent (PGD) (Kurakin et al., 2017) adversaries, which perform multiple updates of (projected) gradient ascent to try to generate a maximally confusing perturbation within some L_p ball of predetermined radius ϵ centred at the chosen data sample.

[1]Mila [2]Université de Montréal [3]University of Cambridge [4]McGill University. Correspondence to: Adam Ibrahim <first.last@mila.quebec>.

2^{nd} *AdvML Frontiers workshop at* 40^{th} *International Conference on Machine Learning*, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Unfortunately, adversarial training can fail to provide high robustness against several attacks, or tunings of attacks, only encountered at test time. For instance, simply changing the norm constraining the search for adversarial examples with PGD has been shown theoretically and empirically (Khoury & Hadfield-Menell, 2018; Tramèr & Boneh, 2019; Maini et al., 2020) to induce significant trade-offs in performance against PGD of different norms. This issue highlights the importance of having a well-defined notion of "robustness": while using the accuracy against individual attacks has often been used as a proxy for robustness, a better notion, as argued by Athalye et al. (2018), is to consider the accuracy against an ensemble of attacks within a threat model (i.e. a predefined set of allowed attacks). Indeed, an attacker will often realistically not be constrained to a single attack, and may attempt several attacks to find one that succeeds.

In order to be robust against multiple attacks, we draw inspiration from domain generalisation. In domain generalisation, we seek to achieve consistent performance even in case of unknown distributional shifts in the inputs at test time. We interpret different attacks as distinct distributional shifts in the data, and propose to leverage existing techniques from the out-of-distribution generalisation literature.

We choose variance REx (Krueger et al., 2021), which consists in using as a loss penalty the variance on the different training domains of the empirical risk minimisation loss. We choose this method as it is conceptually simple, its iterations are no more costly than existing multi-perturbation baselines', it does not constrain the architecture, and it can be used on models pretrained with existing defenses. We consider robustness against an adversary having access to both the model and multiple attacks.

We are interested in the two following research questions:

1. Can REx improve robustness against multiple attacks seen during training?
2. Can REx improve robustness against unseen attacks, that is, attacks only seen at test time?

Our results show that the answer to both questions is yes on the ensembles of attacks used in this work. We show that REx consistently yields benefits across variations in:

datasets, architectures, multi-perturbation defenses, hyper-parameter tuning, attacks seen during training, and attack types or tunings only encountered at test time.

2. Related Work

2.1. Adversarial attacks and defenses

Since the discovery of adversarial examples against neural networks (Szegedy et al., 2014), numerous approaches for finding adversarial perturbations have been proposed (Goodfellow et al., 2015; Madry et al., 2018; Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Croce & Hein, 2020), with the common goal of finding perturbation vectors with constrained magnitude that, when added to the network's input, lead to (often highly confident) misclassification.

A popular defense that has been effective against such attacks is Adversarial Training (Madry et al., 2018), which consists in training on adversarial examples, typically using PGD with L_∞ norm. However, Khoury & Hadfield-Menell (2018) and Tramèr & Boneh (2019) show how training on PGD with a search region constrained by a p-norm may not yield robustness against PGD attacks using other p-norms. One reason is that different radii are typically chosen for different norms, leading to the search spaces of PGD with respect to different norms to potentially have some mutually exclusive regions. Moreover, different attacks, e.g. PGD and the Carlini and Wagner (Carlini & Wagner, 2017) attacks, optimise different losses. As an example, Fig. 1 illustrates how, when training adversarially a model on L_2-norm PGD, the accuracy against one attack may improve while it may decrease against another attack, even for the same p-norm.

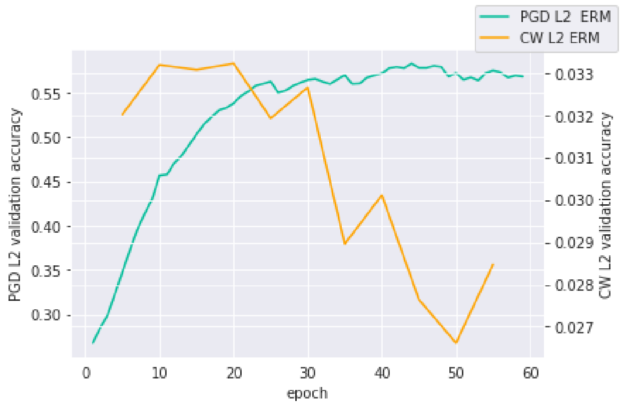
Highlighting the need for methods specific to defending against multiple of perturbations, Tramèr & Boneh (2019) select a set of 3 attacks $\mathcal{A} = \{P_\infty, P_2, P_1\}$, where P_p is PGD with a search region constrained by the L_p norm. They attempt two strategies: the **average (Avg) strategy** consists in training over all attacks in $\mathcal{A}$ for each input (x, y) in the dataset, and the max strategy, which trains on the attack with the highest loss for each sample:

$$L_{\text{Avg}}(\theta, \mathcal{A}) = \mathbb{E}\frac{1}{|\mathcal{A}|}\sum_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{1}$$

$$L_{\max}(\theta, \mathcal{A}) = \mathbb{E}\max_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{2}$$

Maini et al. (2020) propose a modification to the max method: instead of 3 different PGD adversaries that each iterate over their budget of iterations, they design an attack consisting in choosing the worst perturbation among one step of L_∞, L_2 and L_1 PGD every iteration through the chosen number of iterations. This attack, **Multi-Steepest Descent (MSD)**, differs from the max approach of Tramèr & Boneh (2019) where each attack is individually iterated through the budget of iterations first, and the one leading to the worst loss is chosen at the end. Note that this implies that technically, unlike (Tramèr & Boneh, 2019)'s Avg approach, *MSD only consists in training on a single attack.* Maini et al. (2020) show that, in their experimental setup, MSD[1] yields superior performance over Avg and Max.

Figure 1: Validation accuracy of a ResNet18 model adversarially trained on PGD L_2-perturbed CIFAR10, evaluated on PGD and Carlini&Wagner (CW) L_2 attacks.



Nevertheless, there is still a large gap between the performance of such approaches against data perturbed by ensembles of attacks, and the accuracy on the unperturbed data. In order to help address this large gap, we will be exploiting a connection between our goal and domain generalisation.

2.2. Robustness as a domain generalisation problem

Domain generalisation – Out-of-Distribution generalisation (OoD) is an approach to dealing with (typically non-adversarial) distributional shifts. In the domain generalisation setting, the training data is assumed to come from several different domains, each with a different data distribution. The goal is to use the variability across training (or seen) domains to learn a model that can generalise to unseen domains while performing well on the seen domains. In other words, the goal is for the model to have consistent performance by learning to be invariant under distributional shifts. Typically, we also assume access to domain labels, i.e. we know which domain each data point belongs to. Many methods for domain generalisation have been proposed – see (Wang et al., 2021) for a survey.

We frame adversarial robustness as a domain generalisation problem, where the domains stem from different adversarial attacks. Because different attacks use different methods of searching for adversarial examples, and sometimes different search spaces, they may produce different distributions of adversarial examples[2]. One might draw an analogy to

[1]In the rest of the paper, we will use MSD to refer to both the MSD attack, and training on MSD as a defense.

[2]Another way to see this, is that if different attacks or tunings yielded identical distributions, standard results from statistical learning theory would imply similar performance on the attacks.

Hendrycks & Dietterich (2019)'s work on natural pertubations, where both the type and the strength of the perturbations play a similar role as varying the attacks or their tuning, respectively. There are several reasons why the domains we consider may be distributionally shifted with one another (although the distributions may have some overlap). To non-exhaustively name a few, first, we already evoked how different p-norms affect the distributions of adversarial examples yielded by PGD (Khoury & Hadfield-Menell, 2018; Tramèr & Boneh, 2019). Second, different attacks may optimise different losses – for example when comparing P_2 and L_2 CW – which may yield different solutions. Third, the same attack tuned differently (e.g. different ϵ or iteration budget) may yield different distributions of adversarial examples since they do not have the same support. Therefore, robustness to attacks unseen during training means robustness against the corresponding distributional shifts at test time. It is natural to frame adversarial robustness as a domain generalisation problem, as we seek a model that is robust to *any* method to generate adversarially distributional shifts within a threat model, including novel attacks.

In spite of this intuition, it is not obvious that such methods would work in the case of adversarial machine learning. First, recent work demonstrates that domain generalisation methods often fail to improve upon the standard **empirical risk minimisation (ERM)**, i.e. minimising loss on the combined training domains without making use of domain labels (Gulrajani & Lopez-Paz, 2020). On the other hand, success may depend on choosing a method appropriate for the type of shifts at play. Second, a key difference with most work in domain generalisation, is that when adversarially training, the training distribution shifts every epoch, as the attacks are computed from the continuously-updated values of the weights. In contrast, in domain generalisation, the training domains are usually fixed. Non-stationarity is known to cause generalisation failure in many areas of machine learning, notably reinforcement learning (Igl et al., 2020), thereby potentially affecting the success of domain generalisation methods in adversarial machine learning. Third, MSD does not generate multiple domains, which domain generalisation approaches would typically require.

We note that interestingly, the Avg approach can be interpreted as domain generalisation with ERM over the 3 PGD adversaries as training domains. Similarly, the max approach consists in applying a Robust Optimisation approach on the same set of domains. Furthermore, Song et al. (2018) and Bashivan et al. (2021) propose to treat the clean and PGD-perturbed data as training and testing domains from which some samples are accessible during training, and adopt domain adaptation approaches. Therefore, it is difficult to predict in advance how much a domain generalisation approach can successfully improve adversarial defenses.

In this work, we apply the method of **variance-based risk extrapolation (REx)** (Krueger et al., 2021), which simply adds as a loss penalty the variance of the ERM loss across different domains. This encourages worst-case robustness over more extreme versions of the shifts (here, shifts are between different attacks) observed between the training domains, in order to counter adversaries shifting their distribution of attacks to better exploit vulnerabilities in a model. In that light, REx is particularly appropriate given our objective of mitigating trade-offs in performance between different attacks to achieve a more consistent degree of robustness. We note that our implementation of REx has the same computational complexity per epoch as the MSD, Avg and max approaches, requiring the computation of 3 adversarial perturbations per sample.

3. Methodology

Threat model – In this work, we consider white-box attacks, which are typically the strongest type of attacks as they assume the attacker has access to the model and its parameters. Additionally, the attacks considered in the evaluations are gradient-based, with the exception of AutoAttack, which is composite and includes gradient-free perturbations (Croce & Hein, 2020). Because we assume that the attacker has access to all of these attacks, we emphasise that, as argued by Athalye et al. (2018), the robustness against the ensemble of the different attacks is a better metric for how the defenses perform than the accuracy on each individual attack. Thus, using ℓ_{01} as the 0-1 loss, we evaluate the performance on an ensemble of domains $\mathcal{D}$ as:

$$\mathcal{R} = 1 - \mathbb{E} \max_{D \in \mathcal{D}} \ell_{01}(\theta, D(x), y) \tag{3}$$

REx – We propose to regularise the average loss over a set of training domains $\mathcal{D}$ by the variance of the losses on the different domains:

$$L_{\text{REx}}(\theta, \mathcal{D}) = L_{\text{Avg}}(\theta, \mathcal{D}) + \beta \operatorname*{Var}_{D \in \mathcal{D}} \mathbb{E}\, \ell(\theta, D(x), y) \tag{4}$$

where ℓ is the cross-entropy loss. We start penalising by the variance over the training domains once the baseline's accuracies on the seen domains stabilise or peak.

Datasets and architectures – We consider two datasets: MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009). It is still an open problem to obtain high robustness against multiple attacks on MNIST (Tramèr & Boneh, 2019; Maini et al., 2020), even at standard tunings of some commonly used attacks. On MNIST, we use a 3-layer perceptron of size [512, 512, 10]. On CIFAR10, we use the ResNet18 architecture (He et al., 2016). We choose two significantly different architectures to illustrate that our approach may work agnostically to the choice of architecture. We always use batch sizes of 128 when training.

Optimiser – We use Stochastic Gradient Descent (SGD) with momentum 0.9. In subsections 4.2 and B.3 we do not perform hyperparameter optimisation, to isolate the effect of REx from interactions with hyperparameter tuning, which would differ for each defense. We use a fixed learning rate of 0.01 and no weight decay. We fix the coefficient β in the REx loss. In subsection 4.3, we optimise hyperparameters. Based on (Rice et al., 2020) and (Pang et al., 2020), we use in all cases a weight decay of $5 \cdot 10^{-4}$ and a piecewise learning rate decay. For every defense, we search for an optimal epoch to decay the learning rate, with a particular attention to MSD and MSD+REx due to observing a high sensitivity to the choice of learning rate decay milestone. Note that in the case of REx defenses, we always use checkpoints of corresponding baselines before the learning rate is decayed, as we observed this to lead to better performance.

Domains – We consider several domains: unperturbed data, L_1, L_2 and L_∞ PGD (denoted P_1, P_2, P_∞), L_2 Carlini & Wagner (CW$_2$) (Carlini & Wagner, 2017), L_∞ DeepFool (DF$_\infty$) (Moosavi-Dezfooli et al., 2016) and AutoAttack (AA) (Croce & Hein, 2020). We use the Advertorch implementation of these attacks (Ding et al., 2019). For L_∞ PGD, CW and DF, we use two sets of tunings, see appendix A for details. The attacks with a $\bullet$ superscript indicate a harder tuning of these attacks that no model was trained on. Those tunings are intentionally chosen to make the attacks stronger. The set of domains **unseen by all models** is defined as $\{P^\bullet_\infty, DF^\bullet_\infty, CW^\bullet_2, \mathit{AutoAttack}_\infty\}$, with additionally $\mathit{AutoAttack}_2$ in subsection 4.3. The set of domains **unseen by a specific model** is the set of all domains except those seen by the model during training, and therefore varies between baselines. We perform 10 attack restarts per sample to reduce randomness in the test set evaluations.

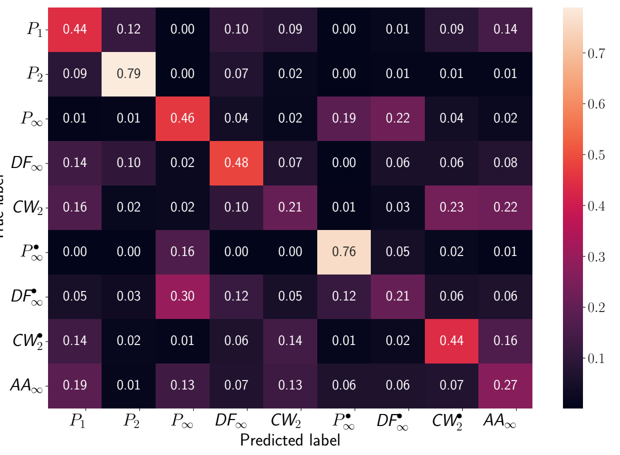
Defenses – Aside from the adversarial training baselines on PGD of L_1, L_2 and L_∞ norms, we define 3 sets of seen domains: $\mathcal{D} = \{\varnothing, P_\infty, DF_\infty, CW_2\}$, $\mathcal{D}_{PGDs} = \{\varnothing, P_1, P_2, P_\infty\}$ and $\mathcal{D}_{MSD} = \{MSD\}$ where $\varnothing$ represents the unperturbed data. We train two Avg baselines: one on $\mathcal{D}$ and one on $\mathcal{D}_{PGDs}$. We train the MSD baseline on $\mathcal{D}_{MSD}$. We use REx on the Avg baselines on the corresponding set of seen domains. However, when REx is used on the model trained with the MSD baseline, we revert to using the set of seen domains $\mathcal{D}_{PGDs}$. While the MSD baseline does not exactly train over P_1, P_2 and P_∞ but rather a composition of these three attacks, we use these attacks when applying REx to the MSD baseline as MSD would only generate one domain, which would not allow us to compute a variance over domains. Note that we chose different sets of seen domains, and different baselines (Avg and MSD), in order to show that REx yields benefits on several multi-perturbation baselines, or within a same baseline with different choices of seen domains. We use **cross-entropy** for all defenses.

See Appendix A for more details about the methodology, such as attack tunings.

4. Results

In this section, we first illustrate the differences in distributions stemming from different families or tunings of attacks by training an attack classifier in subsection 4.1. We then present our results on MNIST and CIFAR10 in subsections 4.2 and 4.3. Additionally, more results on the interaction between the advantage in using REx and hyperparameter tuning, the relative performance of REx models on CIFAR10-C and for transfer learning, and more, can be found in Apdx B. More implementation details, observations and results can generally be found in the appendix (see Table of Content).

Figure 2: Confusion matrix of a discriminator between attacks (normalised by row).



4.1. Attacks as different domains

To illustrate how various types or tunings of attacks may correspond to different distributions, we finetune an ImageNet-pretrained vision transformer (ViT) (Dosovitskiy et al., 2021) to predict which attack was used on the training set of CIFAR10, and test it on the perturbed CIFAR10 test set. Fig. 2 illustrates how without much engineering effort, the ViT is able to tell apart the distribution of perturbations induced by the various attacks with 45.1% accuracy, relative to a random chance of $1/9 \simeq 0.11$. To claim unequivocally that all attacks induce different distributions, we require $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. This is true for all attacks, except two: CW_2 and $DF^\bullet_\infty$.

The cases of CW_2 and $DF^\bullet_\infty$ (the latter sharing $\epsilon = 8/255$ with P_∞ and AA_∞) warrant an additional discussion. The former is due to unsuccessful CW_2 iterations stopping early, i.e. $\tilde{x} \simeq x$. This problem disappears when such unsuccessful CW_2 perturbations are no longer involved. This is why the stronger $CW^\bullet_2$ is classified more easily. As for $DF^\bullet_\infty$ confused with P_∞, we direct the reader to Apdx B.1 for more details, where we show how this problem vanishes when training a binary $DF^\bullet_\infty$ vs P_∞ discriminator.

Table 1: Accuracy on MNIST for different domains. Highlighted cells indicate that the domain (row) was used during training by the defense (column). Bold numbers indicate an improvement of at least 1% accuracy over the baseline used to pretrain REx. Ensembles omit $P_\infty^\bullet$ due to it being overtuned (i.e. tuned to be too strong.).

	Defenses									
	None	Adversarial training			Avg	Avg+REx	Avg$_{PGDs}$	Avg+REx$_{PGDs}$	MSD	MSD+REx
No attack	98.1	98.5	98.3	84.4	99.0	90.0	98.8	87.3	88.4	**90.2**
P_1	95.5	96.8	96.8	44.0	90.3	72.6	95.6	82.5	82.2	**86.8**
P_2	1.8	17.7	63.5	10.0	53.6	44.0	68.3	**72.8**	61.1	**71.8**
P_∞	0.0	0.0	2.2	59.2	67.7	**70.1**	58.0	**70.8**	19.3	**67.4**
DF_∞	3.3	5.7	85.9	78.1	92.9	84.6	92.3	80.9	56.7	**82.4**
CW_2	4.4	6.9	56.5	62.3	68.8	68.3	59.9	41.4	77.1	47.3
$DF_\infty^\bullet$	0.0	0.0	0.0	19.4	7.1	**64.8**	3.7	**58.4**	15.8	**19.9**
$CW_2^\bullet$	2.3	2.8	16.0	30.2	23.2	**42.1**	16.4	12.1	40.2	12.9
AutoAttack$_\infty$	0.0	0.0	0.1	55.0	42.3	**58.8**	34.9	**40.6**	1.5	**31.2**
Ensemble (seen)	-	-	-	-	63.2	63.4	55.5	**64.5**	19.3	**60.1**
Ensemble (unseen by all models)	0.0	0.0	0.0	9.3	3.4	**34.6**	1.2	**8.1**	0.6	**3.9**
Ensemble (unseen by this model)	0.0	0.0	0.0	2.7	3.4	**25.9**	1.2	**8.1**	0.6	**3.9**
Ensemble (all)	0.0	0.0	0.0	2.7	3.4	**25.9**	1.2	**8.1**	0.6	**3.9**
$P_\infty^\bullet$	0.0	0.0	0.0	5.1	0.6	**4.0**	0.9	0.7	0.2	1.0

Key observation 1: Different types or tunings of attacks induce different distributional shifts that a discriminator can identify to some extent, even for the same choice of p-norm and ϵ.

4.2. MNIST

We report our multiperturbation robustness results on MNIST with a multilayer perceptron in Table 1. REx significantly improves robustness against the ensembles of attacks, whether seen or unseen, and in particular on P_∞ and AutoAttack. REx also yields notable improvements against all ensembles, seen or unseen, when used on the Avg baselines. Note however that as in domain generalisation, when used on all baselines except MSD, REx sacrifices performance on the best performing seen domains in order to improve the performance on the strongest attacks. We believe that this trade-off may be worth it for applications where robustness is critical, as for example the 9% of clean accuracy lost by using REx on one Avg baseline translates in an increase of robustness from 3.4% to 25.9% on the ensemble of all attacks excluding the overtuned (i.e. tuned too strongly, leading to negligible accuracy) $P_\infty^\bullet$ adversary.

Our test with tuning the $P_\infty^\bullet$ adversary with $\epsilon = 0.4$ instead of the common tuning of 0.3 on MNIST suggests that REx does not rely on gradient masking(Athalye et al., 2018) compared to the baselines, as the accuracy drops to near 0 values for all models, showing that attacks are successfully computed. This is reinforced by the REx models' AutoAttack performance. A second observation is that the MSD baseline performs surprisingly poorly against AutoAttack and P_∞. We note that experiments with a learning rate schedule (not reported here) did not significantly improve performance of MSD, ruling out the absence of schedule as a cause. While we use the original code of Maini et al. (2020), this could be because we did not use the same architecture as them on MNIST. Furthermore, Maini et al. (2020) did not evaluate on AutoAttack as their work predates the publication of Croce & Hein (2020). In any case, the MSD model did not achieve substantial robustness against P_∞ and AutoAttack in our experiments. This leads to poor performance against all ensembles of attacks, whether seen or unseen, as those include either P_∞ or AutoAttack adversaries. Finally, a third observation is that P_∞ training performs remarkably well in this experiment on the ensemble of attacks, compared to the multi-perturbation baselines.

Key observation 2 (MLP on MNIST): REx improves performance of all baselines on MNIST with a multilayer perceptron, from 3.4% with the best baseline to 25.9% accuracy against an ensemble of L_p attacks, sacrificing a little robustness against the weakest attacks.

4.3. CIFAR10 with hyperparameter optimisation

On CIFAR10 with the ResNet18, we use weight decay and search for an optimal learning rate schedule individually for each defense. The results are summarised in Table 2. We chose not to use the Avg$_{PGDs}$ baseline here, due to performing worse compared to the other baselines on CIFAR10 with a ResNet18 (which we confirm to be especially true when tuning hyperparameters in preliminary experiments). We direct the reader to Apdx B.3 for CIFAR10 results with an ablation on custom hyperparameter optimisation for each model, which leads to even greater advantages in using REx, and which includes the Avg$_{PGDs}$ baseline.

Table 2: Accuracy on CIFAR10, with hyperparameter tuning. Ensembles omit $CW_2^\bullet$ due to overtuning.

	Defenses				
	P_∞	Avg	Avg+REx	MSD	MSD+REx
No attack	80.8	80.0	76.8	78.6	77.4
P_1	78.2	78.3	74.9	76.6	75.2
P_2	70.0	67.9	68.7	69.8	68.7
P_∞	47.3	34.4	**48.1**	45.8	**48.3**
DF_∞	69.0	64.4	**67.1**	67.1	67.3
CW_2	17.4	14.5	**29.6**	17.9	**20.9**
$P_\infty^\bullet$	28.9	16.9	**28.2**	27.4	**30.7**
$DF_\infty^\bullet$	46.3	35.2	**45.3**	44.9	**46.2**
AutoAttack$_\infty$	44.8	33.5	**43.1**	42.8	**44.8**
AutoAttack$_2$	57.7	59.2	58.4	61.1	56.6
Ensemble (seen)	-	14.5	**29.2**	45.8	**48.2**
Ensemble (unseen by all models)	28.9	16.9	**27.9**	27.4	**30.3**
Ensemble (unseen by this model)	16.9	16.9	**27.9**	16.5	**19.6**
Ensemble (all)	16.9	14.2	**23.5**	16.5	**19.6**
$CW_2^\bullet$	2.5	4.8	5.3	1.9	**3.7**

Once again, we observe that REx improves significantly the seen and unseen ensemble accuracies over the baselines. We also note that P_∞ adversarial training performs better than the baselines on the ensemble of attacks used in this paper, even with the addition of AutoAttack$_2$ to the ensembles containing unseen attacks. Moreover, only REx performs better than P_∞ adversarial training on P_∞ attacks. In other words, multi-perturbation defenses only perform better than P_∞ against ensembles of attacks when used with REx.

While MSD performs significantly better with a ResNet18 on CIFAR10 than with the MLP on MNIST (likely due to using the same architecture as them on CIFAR10), as suspected when discussing our optimiser methodology in Sec. 3, there are interaction effects between hyperparameter tuning and the performance of REx relative to a baseline. Improvements of MSD+REx over MSD are sensitive to hyperparameter tuning, specifically at which epoch to start using REx, and when to decay the learning rate. This sensitivity, and the lower advantage of MSD+REx over MSD, is likely due to the fact that the MSD baseline does not train over multiple domains. REx was originally designed to be used with a baseline using ERM on multiple domains as loss function. Therefore, when it is used in tandem with MSD, REx uses the loss indicated in eq. 4. However, because as mentioned before, the Avg$_{PGDs}$ baseline performs significantly worse than the MSD baseline, it is likely that the advantage in using REx is impacted negatively by the suboptimality of the first (ERM) term in the REx loss. Nevertheless, the variance penalty is beneficial enough to achieve higher robustness with MSD+REx than MSD.

Table 3: Accuracy on two non-L_p attacks on CIFAR10.

	Defenses				
	P_∞	Avg	Avg+REx	MSD	MSD+REx
RecolorAdv	50.5	24.5	**63.5**	56.0	**58.2**
StAdv	12.1	4.0	**31.8**	17.6	**22.7**

While our results have focused on L_p attacks, we evaluate the tuned CIFAR10 models on two additional non-L_p attacks: RecolorAdv (Laidlaw & Feizi, 2019), and Spatial Transformations (Xiao et al., 2018). We observe in Table 3 that REx provides significantly better robustness against these perturbations than any other baseline.

Table 4: Average accuracy on CIFAR10-C corruptions.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Average	21.8	52.3	26.5	48.2	42.1	51.2

Additionally, we also report results on CIFAR10-C (Hendrycks & Dietterich, 2019) in Table 4. The dataset consists in mimicking several natural corruptions on CIFAR10 images at various strength, which as argued before, can be seen as a non-adversarial analogue of trying both different types, and tunings of adversarial attacks. While this is not an adversarial robustness benchmark, it shows that REx significantly improves robustness of multiperturbation defenses to non-adversarial shifts it is used on, in spite of what REx models' lower in-distribution clean accuracy on CIFAR10 may have suggested in Table 2.

For more results or details, please consult Apdx B.

Key observations 3 (ResNet18 on CIFAR10):

- REx improves the performance of all baselines on CIFAR10 with a ResNet18, from 16.9% with the best baseline to 23.5% accuracy against an ensemble of L_p attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multi-perturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when they are used in conjunction with REx.
- REx also provides better robustness on some common non-L_p attacks, and significantly improves robustness of baselines it is used on against CIFAR-C's non-adversarial shifts.

5. Conclusion

An attacker seeking to exploit a machine learning model is liable to use the most successful attack(s) available to them. Thus, defenses against adversarial examples should ideally provide robustness against any reasonable attack, including novel attacks. In particular, worst-case performance against the set of available attacks is most reflective of the robustness against a dedicated and sophisticated adversary.

We achieve state-of-the-art worst-case robustness by applying the domain generalisation technique of V-REx (Krueger et al., 2021), which seeks to equalise performance across attacks used at training time. Our approach is simple, practical, and effective. It leads to consistent performance improvements over baselines across different datasets, architectures, training attacks, test attack types and tunings. A limitation, as often in adversarial machine learning, is that our results make no *guarantees* about attacks that were not used in the evaluation. Another limitation lies in the slight loss of accuracy on the unperturbed data, albeit we believe the improvements in adversarial and non-adversarial robustness and promising research directions are significant enough to be of interest to the community. Indeed, our work demonstrates the potential of applying domain generalisation approaches to adversarial robustness. Future work could investigate other OoD generalisation methods such as Distributionally Robust Optimisation (DRO) (Sagawa et al., 2019) or Invariant Risk Minimisation (IRM) (Arjovsky et al., 2019).

References

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv:1907.02893*, 2019.

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pp. 274–283. PMLR, 2018. URL https://arxiv.org/abs/1802.00420.

Bashivan, P., Bayat, R., Ibrahim, A., Ahuja, K., Faramarzi, M., Laleh, T., Richards, B., and Rish, I. Adversarial feature desensitization. *Advances in Neural Information Processing Systems*, 34, 2021. URL https://arxiv.org/abs/2006.04621.

Carlini, N. and Wagner, D. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57. IEEE, 2017. URL https://arxiv.org/abs/1608.04644.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pp. 2206–2216. PMLR, 2020. URL https://arxiv.org/abs/2003.01690.

Ding, G. W., Wang, L., and Jin, X. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019. URL https://arxiv.org/abs/1902.07623.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2010.11929.

Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. URL http://arxiv.org/abs/1412.6572.

Gulrajani, I. and Lopez-Paz, D. In search of lost domain generalization. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2007.01434.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. URL https://arxiv.org/abs/1512.03385.

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *Proceedings of the International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1903.12261.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. The impact of non-stationarity on generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*, 2020. URL https://arxiv.org/abs/2006.05826.pdf.

Khoury, M. and Hadfield-Menell, D. On the geometry of adversarial examples. *arXiv preprint arXiv:1811.00525*, 2018. URL https://arxiv.org/abs/1811.00525.

Kifer, D., Ben-David, S., and Gehrke, J. Detecting change in data streams. In *VLDB*, volume 4, pp. 180–191. Toronto, Canada, 2004.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Zhang, D., Le Priol, R., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). In *International Conference on Machine Learning*, pp. 5815–5826. PMLR, 2021. URL https://arxiv.org/abs/2003.00688.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *ICLR Workshop*, 2017. URL https://arxiv.org/abs/1607.02533.

Laidlaw, C. and Feizi, S. Functional adversarial attacks. *Advances in neural information processing systems*, 32, 2019. URL https://arxiv.org/abs/1906.00001.

Laidlaw, C., Singla, S., and Feizi, S. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2006.12655.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Maini, P., Wong, E., and Kolter, Z. Adversarial robustness against the union of multiple perturbation models. In *International Conference on Machine Learning*, pp. 6640–6650. PMLR, 2020. URL https://arxiv.org/abs/1909.04068.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. 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, 2016. URL https://arxiv.org/abs/1511.04599.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011. URL http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2010.00467.pdf.

Rice, L., Wong, E., and Kolter, Z. Overfitting in adversarially robust deep learning. In *International Conference on Machine Learning*, pp. 8093–8104. PMLR, 2020. URL https://arxiv.org/abs/2002.11569.pdf.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization. *arXiv:1911.08731*, 2019. URL https://arxiv.org/abs/1911.08731.

Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., and Madry, A. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020. URL https://arxiv.org/abs/2007.08489.

Singh, M., Gustafson, L., Adcock, A., de Freitas Reis, V., Gedik, B., Kosaraju, R. P., Mahajan, D., Girshick, R., Dollár, P., and Van Der Maaten, L. Revisiting weakly supervised pre-training of visual perception models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 804–814, 2022. URL https://arxiv.org/abs/2201.08371.

Song, C., He, K., Wang, L., and Hopcroft, J. E. Improving the generalization of adversarial training with domain adaptation. *arXiv preprint arXiv:1810.00740*, 2018. URL https://arxiv.org/abs/1810.00740.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

Tramèr, F. and Boneh, D. Adversarial training and robustness for multiple perturbations. *arXiv preprint arXiv:1904.13000*, 2019. URL https://arxiv.org/abs/1904.13000.

Utrera, F., Kravitz, E., Erichson, N. B., Khanna, R., and Mahoney, M. W. Adversarially-trained deep nets transfer better: Illustration on image classification. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2007.05869.

Wang, J., Lan, C., Liu, C., Ouyang, Y., Zeng, W., and Qin, T. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021. URL https://arxiv.org/abs/2103.03097.

Xiao, C., Zhu, J.-Y., Li, B., He, W., Liu, M., and Song, D. Spatially transformed adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://arxiv.org/abs/1801.02612.

Contents

The code can be found at https://github.com/AIproj/Towards-Out-of-Distribution-Adversarial-Robustness.

A. More on methodology

A.1. Motivation

Benchmark design: we use a similar benchmark as (Tramèr & Boneh, 2019) and (Maini et al., 2020), with the addition of AutoAttack. Furthermore, to empirically support our claims that REx improves worst case robustness, we additionally consider ensembles of seen, unseen, and all attacks. We consider worst-case ensembles because for real-world robustness, an attacker may try several attacks until they succeed. The evaluation against an ensemble of seen attacks is to support our first claim, i.e. that REx improves multi-perturbation robustness, and the consideration of unseen attacks supports the second claim: that REx generalises to attacks that were not seen during training.

Why weak attacks: weak attacks capture a signal that would be missed otherwise. Indeed, as argued in the discussion of our results, REx does improve significantly robustness against strong attacks and even unbounded attacks (which can't be ignored in the general context of robustness until out-of-distribution detectors are perfectly able to capture shifts beyond the typical balls); however, REx often lowers the accuracy slightly on the weakest attacks. In other words, weak attacks allow us to capture a trade-off in using REx, which is important information for readers. Not having those evaluations would suggest that REx is an improvement in every setting.

Unbounded attacks: most work in the literature focuses on bounded attacks. This is because bounded attacks constrain the strength of attacks, and have long been needed to obtain non-trivial robustness results. In this work, we consider several of the most commonly used bounded attacks, and show the improvement yielded by REx on the ones affecting the accuracy of the model the most. However, at deployment, an attacker might use some unbounded attacks, especially if they are not perceptible. We encourage the reader to decide from Fig. 10 and 11 whether they consider the corruption to be perceptible by a human, in spite of being out of the $\epsilon = 0.5$ L_2 ball. In the absence of algorithms able to detect perturbations larger than the usual choices of ϵ, we argue that non-visually perceptible adversarial attacks, regardless of being bounded or not, are of particular concern. Therefore, we additionally consider an unbounded attack (CW) to show that REx provides benefits even in that less-studied case. Note that we find that REx also improves robustness when bounding CW by rejecting examples out of the L_2 ball of radius $\epsilon = 0.5$ on CIFAR10.

Keeping "overtuned" attacks: in our tables of results, we refer to some attacks are "overtuned". What is meant is that after choosing an alternative tuning of those attacks, all models mostly failed against them. For example, for $P_\infty^\bullet$ on MNIST, this is done by choosing an $\epsilon = 0.4$ instead of the commonly used value of 0.3. Since we already provide a significant number of evaluations throughout this work, we decided to report those overtuned attacks to highlight two points. First, the models still have weakness, and as seen in Fig. 10 and 11, adversarial examples produced by the overtuned attacks are not necessarily perceptible, so a truly robust model should defend against them. Second, as argued in the discussion, these attacks, along with the AutoAttack evaluation, highlight that REx does not rely on gradient obfuscation (Athalye et al., 2018).

Ablation on hyperparameter optimisation: we perform an ablation on hyperparameter optimisation on CIFAR10 in Sec. B.3. This is done in order to highlight several points. First, that without hyperparameter optimisation, REx-based defenses may outperform hyperparameter-optimised baselines. Second, as argued in the discussion of Sec. 4.3, in the specific case of the MSD baseline, while the use of REx on that baseline still yields improvements over the baseline, it is slightly less pronounced than without hyperparameter optimisation.

Max baseline: we choose not to include the Max baseline from (Tramèr & Boneh, 2019) since Tables 3 and 4 of (Maini et al., 2020) have evaluations on a very similar benchmark. Note that Maini et al. (2020) find on CIFAR10 that Avg performs significantly better than Max. Moreover, on both MNIST and CIFAR10, Maini et al. (2020) find that MSD performs better than both Avg and Max anyway, so improving on MSD with REx implies that MSD+REx would outperform Max on both datasets.

A.2. Attack tunings

Using Advertorch's and Croce's implementation of AutoAttack's[3] parameter names, we report the attacks' tuning here.

A.2.1. MNIST (SUBSECTION 4.2)

1. P_1: $\epsilon = 10$, $n_{iter} = 40$, $\epsilon_{iter} = 0.5$
2. P_2: $\epsilon = 2$, $n_{iter} = 40$, $\epsilon_{iter} = 0.1$
3. P_∞: $\epsilon = 0.3$, $n_{iter} = 40$, $\epsilon_{iter} = 0.01$
4. DF_∞: $\epsilon = 0.11$, $n_{iter} = 30$
5. CW_2: *max_iterations* $= 20$, *learning_rate* $= 0.1$, *binary_search_steps* $= 5$
6. $P_\infty^\bullet$: $\epsilon = 0.4$, $n_{iter} = 40$, $\epsilon_{iter} = 0.033$
7. $DF_\infty^\bullet$: $\epsilon = 0.4$, $n_{iter} = 50$
8. $CW_2^\bullet$: *max_iterations* $= 30$, *learning_rate* $= 0.12$, *binary_search_steps* $= 7$
9. $AutoAttack_\infty$: $\epsilon = 0.3$, *norm* $=$ "Linf"

The MSD attack uses the same tuning as the individual P_p attacks.

A.2.2. CIFAR10 WITHOUT HYPERPARAMETER OPTIMISATION (SUBSECTION B.3)

1. P_1: $\epsilon = 10$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
2. P_2: $\epsilon = 0.5$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
3. P_∞: $\epsilon = \frac{8}{255}$, $n_{iter} = 40$, $\epsilon_{iter} = \frac{2}{255}$
4. DF_∞: $\epsilon = 0.011$, $n_{iter} = 30$
5. CW_2: *max_iterations* $= 20$, *learning_rate* $= 0.01$, *binary_search_steps* $= 5$
6. $P_\infty^\bullet$: $\epsilon = \frac{12}{255}$, $n_{iter} = 70$, $\epsilon_{iter} = \frac{2}{255}$
7. $DF_\infty^\bullet$: $\epsilon = \frac{8}{255}$, $n_{iter} = 50$
8. $CW_2^\bullet$: *max_iterations* $= 30$, *learning_rate* $= 0.012$, *binary_search_steps* $= 7$
9. $AutoAttack_\infty$: $\epsilon = \frac{8}{255}$, *norm* $=$ "Linf"

MSD uses the same tuning as the individual P_p attacks.

A.2.3. CIFAR10 WITH HYPERPARAMETER OPTIMISATION (SUBSECTION 4.3)

We use the same tuning as above for testing, with the addition of an $AutoAttack_2$ adversary with $\epsilon = 0.5$ and *norm* ="L2". However, for training, based on (Rice et al., 2020), we set

1. P_1: $\epsilon = 10$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{20}{255}$
2. P_2: $\epsilon = 0.5$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{15}{255}$
3. P_∞: $\epsilon = \frac{8}{255}$, $n_{iter} = 10$, $\epsilon_{iter} = \frac{2}{255}$

and do the same for MSD.

[3] https://github.com/fra31/auto-attack

A.3. More on how the attack discriminator is trained

In order to train the attack discriminator of subsections 4.1 and B.1, we load the “IMAGENET1K_SWAG_LINEAR_V1” weights (Singh et al., 2022) of a ViT b16 available on Torchvision with

```
torchvision.models.vit_b_16(weights="IMAGENET1K_SWAG_LINEAR_V1")
```

We freeze all but the last (linear) layer of the ImageNet-pretrained ViT, which we reset and adapt to the proper number of output classes, and which is therefore the only layer that we train. We use a ViT because it has significantly higher performance than a ResNet18, which we tried initially and found to have much difficulty converging to good classifiers both on the validation and test sets, even after 150 epochs. In contrast, the ViT almost reaches its final values in 2 epochs.

We proceed by using the attacks with their tuning from subsubsection A.2.3 on CIFAR10, using the hyperparameter-optimised model adversarially trained against P_∞. Since the task is to show that a discriminator can classify different attacks, we do not need to perturb all CIFAR10 classes and limit ourselves to a single at a time. The confusion matrices provided in this work are based on perturbing all CIFAR10 samples of class "0" (airplanes). After saving all the images of airplanes perturbed with every attack, we load them as a dataset for attack classification with the pretrained ViT. Our code allows one generate or use this dataset, selecting the classes and attacks used.

As preprocessing, we transform the images into perturbations by subtracting the unperturbed original image from the adversarial images, i.e. $A(x^0) - x^0$ where x^0 is the original unperturbed image and A is an attack. Furthermore, we opt to standardise at an instance level (i.e. per image) instead of using batch or dataset statistics. A seed of 0 is used in every case.

We use Adam (Kingma & Ba, 2014) as an optimiser with its default Pytorch settings in torch1.13[4] (lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0).

We also report the epoch at which we early stopped training for the analyses:

1. Fig. 2 (discriminating between all L_p attacks considered in this work): epoch 40.
2. Fig. 3 (discriminating between P_∞ and $DF^{\bullet}_{\infty}$): epoch 16.
3. Fig. 4 (discriminating between $P_1, P_2, P_\infty, DF_\infty, CW_2$): epoch 56.

As a reminder, the same base dataset is used for all 3 models, retaining only the samples corresponding to attacks selected for the analysis.

Disclaimer: we made no particular attempt to improve further the attack discriminator. It is probable that more engineering, e.g. transfer learning or preprocessing heuristics, might lead to even better results. However, we believe our current models achieve the intended goal of showing that all these attacks induce different distributions even for the same p-norm and $epsilon$, as evidenced by the fact that for any attack, we can show on the test set that $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. See 4.1 and B.1.

A.4. More on how the models are trained

Note that when activating REx, we always reset the optimiser to avoid using accumulated momentum from the baseline. When tuning hyperparameters, we find that activating the REx penalty after learning rate decays generally is a worse strategy than activating it before when the baseline’s accuracy decreases after a decay. All models are trained using a single NVIDIA A100 for MNIST and 2 NVIDIA A100s for CIFAR10.

A.4.1. No hyperparameter optimisation

First, we pretrain the architecture on the clean dataset. Then, the baseline is trained on the appropriate seen domains. On MNIST, convergence does not happen in many baselines’ case until thousands of epochs. Therefore, we choose to stop training when progress on the seen domains slows, as in Fig. 5 where we stopped training for example at epoch 1125 for

[4] Every other experiment uses torch1.8.1, the ViT analysis being a late addition to this work and ViT requiring more recent Pytorch versions.

both the Avg and the Avg+REx models. For CIFAR10, as Fig. 6 indicate, the accuracies peak in significantly less epochs, so we early stop when the Ensemble (seen) accuracy on the seen domains peaks. Note that we do this manually by looking at the seen domains' validation curves (see Sec. C for more details on early stopping). REx is triggered on a baseline before the baseline's early stopping epoch, when progress on the seen domains slows.

An important precision about Fig. 5, 6 is that unseen attack performance is only evaluated every 5 epochs, hence the jagged aspect of the curves. We do this because of the huge computational cost of running all 9 attacks on each sample every epoch.

For a full description of early stopping and when we activated the REx penalty on baselines:

MNIST

- P_1 model: early stopped at epoch 95
- P_2 model: early stopped at epoch 75
- P_∞ model: early stopped at epoch 1125
- Avg model: early stopped at epoch 1125
- Avg+REx model: REx penalty activated at epoch 726, early stopped at epoch 1125
- Avg$_{PGDs}$ model: early stopped at epoch 1105
- Avg+REx$_{PGDs}$ model: REx penalty activated at epoch 551, early stopped at epoch 1105
- MSD model: early stopped at epoch 655
- MSD+REx model: REx penalty activated at epoch 101, early stopped at epoch 655

CIFAR10

- P_1 model: early stopped at epoch 69
- P_2 model: early stopped at epoch 59
- P_∞ model: early stopped at epoch 45
- Avg model: early stopped at epoch 50
- Avg+REx model: REx penalty activated at epoch 301, early stopped at epoch 330
- Avg$_{PGDs}$ model: early stopped at epoch 95
- Avg+REx$_{PGDs}$ model: REx penalty activated at epoch 301, early stopped at epoch 370
- MSD model: early stopped at epoch 40
- MSD+REx model: REx penalty activated at epoch 26, early stopped at epoch 70

A.4.2. WITH HYPERPARAMETER OPTIMISATION

The experiments are run with a ResNet18 architecture on CIFAR10. We follow the results of Rice et al. (2020) and Pang et al. (2020), using a piecewise learning rate schedule decay and a weight decay value of $5 \cdot 10^{-4}$. In all cases we start with a learning rate of 0.1, decayed to 0.01 at the corresponding milestone. In preliminary tuning experiments, we observe the Avg$_{PGDs}$ to perform very poorly relative to other baselines, and chose to drop that baseline.

- P_∞ model: milestone at epoch 100, early stopped at epoch 103
- Avg model: milestone at epoch 100, early stopped at epoch 140

- Avg+REx model: REx penalty activated at epoch 50 (before lr decay), milestone at epoch 100, early stopped at epoch 110
- MSD model: milestone at epoch 50, early stopped at epoch 51
- MSD+REx model: REx penalty activated at epoch 50 (before lr decay), milestone at epoch 97, early stopped at epoch 99

We attempt several learning rate schedule milestones for both MSD and MSD+REx, which has a higher impact than for other models. This process can be automated since the worst-case seen accuracy always peaks within a few epochs of a milestone.

A.5. Other implementation details

We use the implementation of `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` for ResNet18. For StAdv and RecolorAdv, we use the code of Laidlaw & Feizi (2019), available at `https://github.com/cassidylaidlaw/ReColorAdv`.

REx's β parameter is generally set to 10, except for MSD+REx on MNIST in subsection 4.2 where it is set to 4. These numbers initially come from setting β to a value of the same order of magnitude as $\frac{L_{Avg}}{\text{Var}}$'s value at the epoch REx is activated, in the early iterations of our experiments. This is done to encourage the optimisation dynamics to neglect neither term of the REx loss. We found that $\beta = 10$ worked generally well, even in many settings where empirically $\frac{L_{Avg}}{\text{Var}} \simeq 30$. See Sec. D for a discussion of how the choice of β affects performance.

B. Additional results

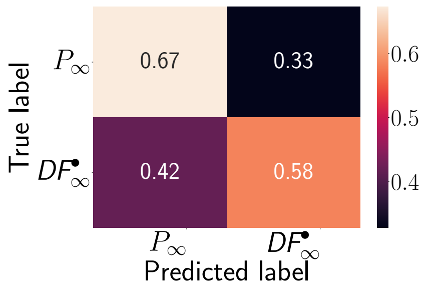
B.1. More about the attack discriminator

We invite the reader to consult Apdx A.3 for details about the methodology of the results presented here and in subsection 4.1.

Here, we give more results about the attack discriminator. To claim unequivocally that all attacks considered are distributionally shifted with respect to one another, we require $p(A_{true}) > p(A \neq A_{true})$ where A_{true} is the true attack used to generate the sample and $p(A)$ is the probability predicted by the model that a perturbation corresponds to attack A. [5] This is true for all attacks as seen in Fig. 2, except $DF^{\bullet}_{\infty}$ being confused for P_{∞}, and CW_2 being confused for $CW^{\bullet}_2$ or AutoAttack$_{\infty}$.

$DF^{\bullet}_{\infty}$ and P_{∞} share the same $\epsilon = 8/255$ and the model on which the attacks were generated (the P_{∞} adversarially trained model from Table 2) performs roughly equally on both (46.3% accuracy on $DF^{\bullet}_{\infty}$ vs 47.3% on P_{∞}). Therefore, it would be particularly important for our claim to be able to measure some difference between those distributions, even if there may be some overlap. Note that despite sharing a similar performance and ϵ, AutoAttack$_{\infty}$ does not induce the same confusion.

Figure 3: Confusion matrix of ViT predicting whether perturbations stem from P_{∞} or $DF^{\bullet}_{\infty}$ (normalised by row).



In order to tackle this, we train a binary ViT classifier to tell apart $DF^{\bullet}_{\infty}$ and P_{∞}. We can see in Fig. 3 that the model is now able to properly predict each class with higher probability than the other, with average probability 63%. The confusion (off-diagonal entries) are indicative that as we might suspect, there is *some* overlap between the distributions of perturbations generated by the two attacks.

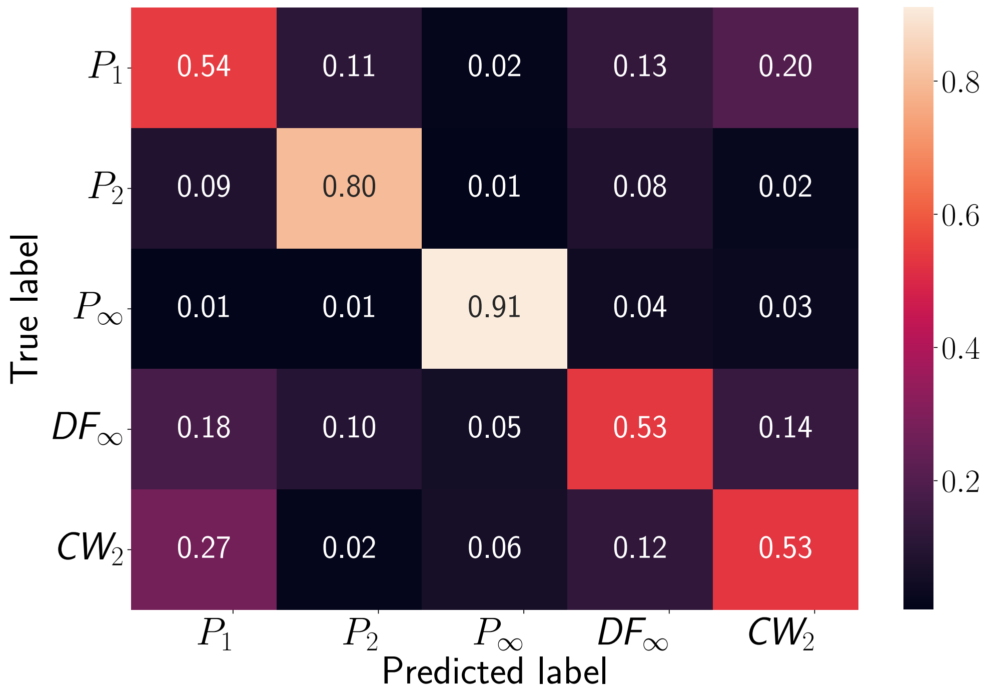
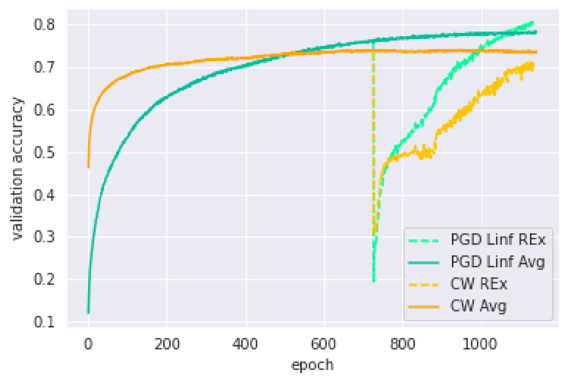


Figure 4: Confusion matrix of ViT predicting whether perturbations stem from 5 different types of attacks (normalised by row). Test accuracy is 66.3%.

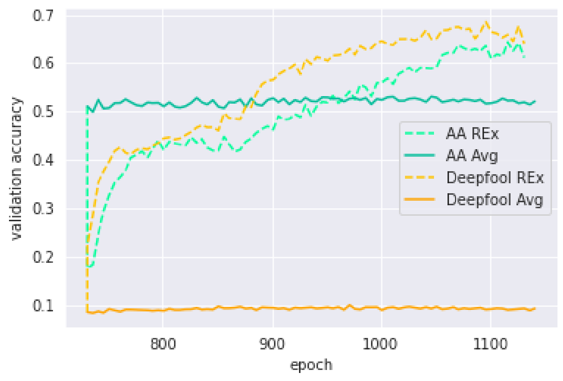
Concerning the CW_2 confusion results, this is an artefact of the implementation of CW_2 that we use. Indeed, the implemen-

[5] While this is out of the scope of this work, it is possible to relate the divergence between the distributions to the performance of the discriminator, as done by Bashivan et al. (2021) using Kifer et al. (2004)'s $\mathcal{H}\Delta\mathcal{H}$ divergence theory.

tation of the attack from advertorch early stops by default if the iterations get stuck in a local minimum. This leads to some samples having near 0 perturbation and not being adversarial. As argued in the main text, this is no longer an issue as soon as CW_2 is tuned so that this does not happen, e.g. with the $CW_2^{\bullet}$ tuning.



(a) MNIST seen attacks



(b) MNIST unseen attacks

Figure 5: Validation accuracy of Avg on MNIST with and without REx (dashed line), against seen attacks (left) and unseen attacks (right) (AA=AutoAttack).

B.2. Early stopping and REx with MNIST

In Figure 5, we observe how regularising by the variance over the domains leads to better peak performance when using REx, over different baselines. Like the baselines, we early stop REx models. For all defenses, we use the validation set to choose when to early stop, by selecting the epoch when the performance peaks on the ensemble of seen domains. As shown on the curves, even though we stop training before REx reaches higher performance on the seen attacks, we still get significant improvements on the individual unseen attacks and against the ensembles, as reported in Table 1. See Appendix A for more details on how REx is used, and Appendices B.3 and C for more details on how, in contrast with MNIST with the MLP, early stopping is required much earlier and the validation curves no longer behave monotonically on a different dataset and architecture.

B.3. CIFAR10 - no hyperparameter optimisation

The results on CIFAR10 without performing hyperparameter optimisation on each model are summarised in Table 5. We observe again that REx is an improvement over the ensemble of seen attacks compared to the baselines it was used on. As on MNIST, this happens by improving the performance on the strongest of the seen attacks and sacrificing a little performance on the top performing attack(s). Moreover, REx consistently yields a significant improvement in robustness when evaluated against the ensemble of unseen attacks, too. The only individual domains where REx never yields significant improvements are the clean (unperturbed) data, P_1 and P_2, whether they were seen during training or not. Given the relatively good performance of the baselines and REx on those domains, this is in line with REx's tendency to sacrifice a little accuracy on the best performing domains to improve significantly the performance on the worst performing domains.

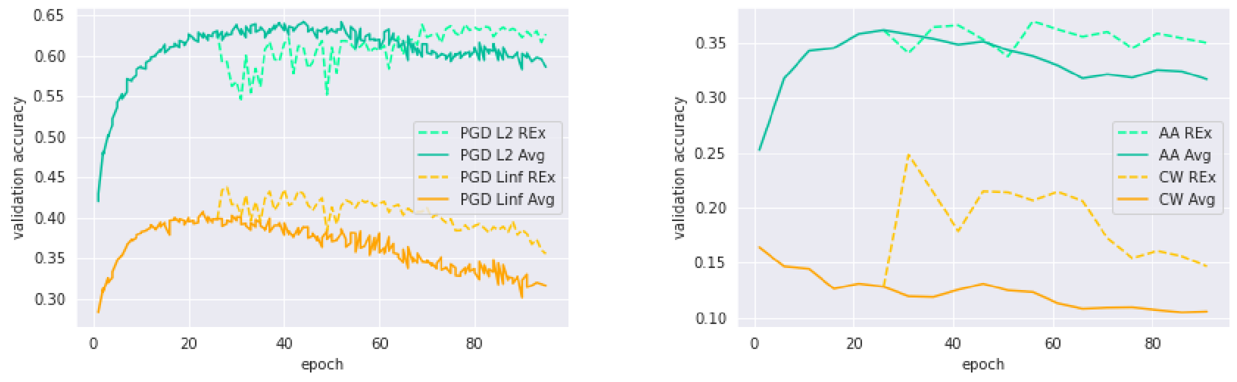
While adversarial training on either P_1 or P_2 fails to yield robustness to unseen attacks, we observe that these two defenses are the only ones for which the clean accuracy does not decrease significantly. We note that unlike on MNIST, MSD is significantly more competitive with the other baselines, and its performance is relatively similar to the one reported by Maini et al. (2020) (likely due to using the same architecture on CIFAR10).

As with tuned models (subsec. 4.3), the model trained on P_∞ performs better than the Avg, Avg$_{PGDs}$ and MSD models on the set of attacks unseen by all models. Conversely, training on ensembles of attacks also hurts performance on P_∞, unless we apply REx. In other words, only REx is able to improve both P_∞ *and* worst-case performance over an ensemble of attacks. Moreover, without hyperparameter tuning, REx appears to lose more performance on the best performing domains. This is particularly notable in the case of P_2 attacks, where for example REx improves the P_2 accuracy by $+0.8\%$ with hyperparameter tuning, vs -4.5% without with the Avg model training on $\{P_\infty, DF_\infty, CW_2\}$. In contrast, the gap in performance in performance between MSD and MSD+REx is even larger than without individual hyperparameter

Table 5: Accuracy on CIFAR10 for different domains. Ensembles omit $CW_2^\bullet$ due to overtuning.

	Defenses									
	None	Adversarial training			Avg	Avg+REx	Avg$_{PGDs}$	Avg+REx$_{PGDs}$	MSD	MSD+REx
No attack	87.6	92.1	87.1	77.4	80.9	75.1	82.8	79.0	76.3	75.1
P_1	80.3	87.6	85.0	75.7	78.7	72.0	80.4	77.0	74.4	72.7
P_2	19.9	47.8	70.9	66.6	69.8	65.3	71.1	68.0	65.7	65.9
P_∞	0.0	0.0	9.7	39.5	34.4	**44.2**	32.3	**41.2**	37.9	**41.9**
DF_∞	4.1	19.2	60.2	64.6	64.9	62.2	66.8	64.5	62.7	63.6
CW_2	0.0	0.0	1.3	11.2	9.8	**21.6**	8.7	**16.5**	10.7	**17.5**
$P_\infty^\bullet$	0.0	0.0	1.0	20.1	16.3	**24.1**	13.2	**22.1**	19.3	**23.8**
$DF_\infty^\bullet$	0.0	0.0	9.5	38.5	35.6	**40.5**	33.0	**39.1**	36.6	**40.4**
AutoAttack$_\infty$	0.0	0.0	8.1	37.2	33.7	**38.8**	31.2	**37.6**	36.0	**39.0**
Ensemble (seen)	-	-	-	-	9.8	**21.2**	32.3	**41.2**	37.9	**41.8**
Ensemble (unseen by all models)	0.0	0.0	1.0	20.1	16.3	**24.0**	13.2	**22.0**	19.3	**23.6**
Ensemble (unseen by this model)	0.0	0.0	0.9	10.7	16.3	**24.0**	7.7	**14.2**	10.3	**16.1**
Ensemble (all)	0.0	0.0	0.9	10.7	9.4	**17.9**	7.7	**14.2**	10.3	**16.1**
$CW_2^\bullet$	0.0	0.0	0.1	1.1	1.6	**2.6**	1.1	**2.7**	1.0	**2.7**

optimisation, showing that REx helps bridge gaps in performance due to suboptimal tuning.



(a) CIFAR10 seen attacks (b) CIFAR10 unseen attacks

Figure 6: Validation accuracy of MSD on CIFAR10 (bottom) with and without REx (dashed line), against seen attacks (left) and unseen attacks (right) (AA=AutoAttack).

As with MNIST and the CIFAR10 hyperparameter-optimised models, early stopping is important, and moreso than with MNIST, the performance of REx performance quickly peaks, as seen in Fig. 6.

Key observations 4 (no hyperparameter tuning):

- REx improves the performance of all baselines on CIFAR10 with a ResNet18, from 10.7% with the best baseline to 17.9% accuracy against an ensemble of attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multi-perturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when using REx.
- REx helps bridge gaps in performance due to suboptimal hyperparameter tunings.

B.4. Perceptual Adversarial Training and additional experiments

Perceptual Adversarial Training: due to the similarity of our motivation with that of Perceptual Adversarial Training (PAT) (Laidlaw et al., 2021) (i.e. be robust against imperceptible attacks), we evaluate their model on our benchmark. We simply run the code made publicly available by the authors in order to train with PAT a model, and load it in our evaluation pipeline. With a ResNet18 on CIFAR10, we find that they achieve considerably worse robustness, with Ensemble (all) worst-case robustness of 3.8%. The model is noticeably weak against L_∞ attacks, achieving 6.9% accuracy on AutoAttack and 10.0% on P_∞. While this may be surprising, this can be explained by noting that Laidlaw et al. (2021) use a ResNet50 on CIFAR10, while we used their code with the "–arch resnet18" argument to train a model immediately comparable to the main results of our work.

Robustness against different tunings of AutoAttack: we consider an additional tuning of AutoAttack L_∞ on CIFAR10 with the hyperparameter-optimised ResNet18, this time with $\epsilon = 12/255$. We find that the MSD model achieves 25.0% accuracy while MSD+REx achieves 26.8%. The Avg model from Table 2 achieves 15.6% accuracy while its REx counterpart reaches 24.2% in 30 less epochs of training (see Sec. A.4.2 for number of epochs for each model). The P_∞ model yields 25.9%, which is higher than the other baselines, but still worse than MSD+REx. This further supports the claim that REx improves robustness against unseen attacks and tunings of attacks.

B.5. Additional results on CIFAR10-C

Table 6: Accuracies of tuned CIFAR10 models on CIFAR10-C corruptions.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
brightness	31.7	61.5	29.1	54.5	51.7	60.2
contrast	35.4	43.7	33.4	40.6	39.8	42.7
defocus blur	22.9	52.6	25.8	49.1	44.9	52.8
elastic transform	21.6	50.1	24.5	46.4	42.1	50.3
fog	32.5	45.8	32.7	44.2	41.7	45.8
frost	28.4	63.1	33.6	56.8	54.5	61.0
gaussian blur	22.2	51.4	25.7	48.1	44.5	52.0
gaussian noise	13.4	53.3	25.1	48.7	35.4	50.9
glass blur	18.0	49.7	24.2	46.1	39.8	48.3
impulse noise	14.7	47.7	24.8	45.6	33.0	46.0
jpeg compression	18.6	54.7	25.4	50.5	44.8	53.7
motion blur	23.2	49.6	24.9	46.3	42.9	50.4
pixelate	20.9	54.8	25.3	50.2	44.6	53.2
saturate	20.8	49.1	20.8	43.6	37.6	47.6
shot noise	13.6	52.3	24.8	48.4	34.5	50.3
snow	22.3	57.9	27.6	52.4	47.9	55.8
spatter	18.4	52.6	24.6	48.4	42.4	50.5
speckle noise	13.2	50.9	24.5	47.5	32.3	49.0
zoom blur	22.3	52.3	26.1	48.1	45.2	52.7
average	21.8	52.3	26.5	48.2	42.1	51.2

In order to test the investigate the robustness of the different tuned baselines against non-adversarial perturbations, we evaluate them on the corruptions of CIFAR10-C (Hendrycks & Dietterich, 2019). Table 6 shows how P_∞ is the overall best performing model. The Avg and MSD baselines perform surprisingly poorly on CIFAR10-C, and REx leads to very sigificant improvements on both baselines that bring the performance closer to that of the model trained on P_∞. In particular, MSD+REx comes very close (within 2% accuracy) to the P_∞ baseline's performance, only outperforming P_∞ marginally on defocus blurs, elastic transforms, gaussian blurs, motion blurs, and zoom blurs, and having equal performance on fog. This is in contrast with the intuition that because REx had lower (in-distribution) clean accuracy than most other baselines on CIFAR10, this would also be true for out-of-distribution non-adversarial data.

B.6. Transferability of REx models

Several works have highlighted how adversarial robustness can be of interest for transfer learning (Salman et al., 2020; Utrera et al., 2021). We freeze the parameters of the hyperparameter-optimised CIFAR10 models, only resetting the last linear layer (and adapting it to the appropriate number of output classes) and allowing it to train on the CIFAR100 (Krizhevsky et al., 2009) and SVHN (Netzer et al., 2011) training datasets, then evaluating them on their test sets. We train for 30 epochs with a learning rate of 0.1, decaying by factors 0.1 at epochs 10 and 20. We repeat this 3 times per model, using a different seed ("0", "1" and "2") to report average accuracies on the test set and standard deviations.

Disclaimer: we did not attempt any particular optimisation of these results with state of the art techniques, and merely provide these results for completeness. We also highlight that work on transfer learning with robust models often focuses on larger models (e.g. ResNet50 for Utrera et al. (2021)), which can significantly affect the results. Finally, we realise that transferring from CIFAR10 to CIFAR100 (instead of the other way around) may be overly ambitious, with works such as (Utrera et al., 2021) focusing instead on CIFAR100 to CIFAR10.

Table 7: Accuracies of tuned CIFAR10 models finetuned on CIFAR100, averaged over 3 finetunings with different seeds per defense.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Accuracy	28.9±0.1	29.3±0.1	25.9±0.2	27.7±0.2	28.4±0.1	27.7±0.1

On CIFAR100, we can see on Table 7 that the P_∞ model transfers best, marginally better than the non-robust model. All other baseline perform worse than the non-robust model, and REx only appear to improve the Avg baseline, while decreasing by 0.7% the accuracy when used on the MSD baseline before the finetuning. The performance of the non-robust model is perhaps not surprising, given the similarity between the CIFAR10 and CIFAR100 classes, which may explain the relative performances of the baselines being correlated with those observed on CIFAR10 on unperturbed data.

Table 8: Accuracies of tuned CIFAR10 models finetuned on SVHN, averaged over 3 finetunings with different seeds per defense.

	Defenses					
	None	P_∞	Avg	Avg+REx	MSD	MSD+REx
Accuracy	42.4±0.2	49.2±0.3	44.8±0.1	47.1±0.1	49.1±0.2	48.0±0.1

In order to investigate this hypothesis, we also attempt a similar finetuning on the SVHN dataset. The (cropped) SVHN dataset consists in pictures of house numbers, where the task is to read the digit at the center of the image. Therefore, this task corresponds to a shift substantially different than CIFAR100. As suspected, we observe in Table 8 than all of the robust models now have significantly better performance than the non-robust model. The P_∞ and MSD models both transfer the best, with the MSD+REx model losing again roughly 1% of performance compared to MSD, but still ahead of the non-robust model or the Avg and Avg+REx models.

C. When to early stop, and how learning rate milestones affect performance

This section shows results about learning rate schedule milestones and early stopping for the CIFAR10 models with hyperparameter optimisation.

C.1. MSD model

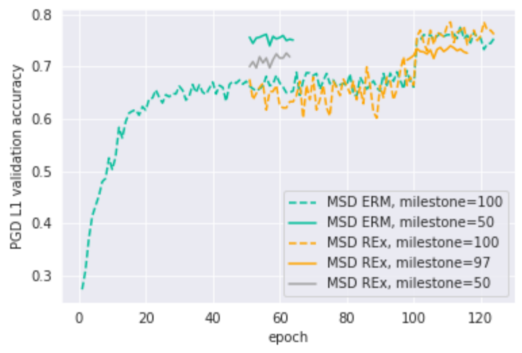
In this subsection, we illustrate two points using Fig. 7: how early stopping is performed on the MSD and MSD+REx models, and how the choice of learning rate decay milestone affects performance. Regarding the former, we early stop based on the peak of the validation Ensemble (seen) accuracy. We motivated that worst-case performance is a more appropriate notion of robustness, and unseen attacks should not be used to make model-selection decisions as they are used to simulate "future", novel attacks that were not known when designing the defenses. Concerning early stopping, we note that the peak performance on Ensemble (seen) accuracy is reached:

- At epoch 101 for the MSD model with milestone at epoch 100
- At epoch 51 for the MSD model with milestone at epoch 50
- At epoch 105 for the MSD+REx model with milestone at epoch 100
- At epoch 99 for the MSD+REx model with milestone at epoch 97
- At epoch 54 or 56 for the MSD+REx model with milestone at epoch 50.

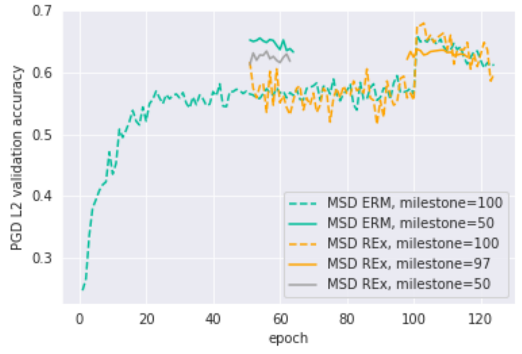
Regarding how milestone choice affects performance, this illustrates how we searched for hyperparameters. We attempt learning rate decays milestones at various epochs, which we then evaluate only for a few epochs, as the performance decays fast anyway as predicted by Rice et al. (2020). The curves in Fig. 7 represent the best learning rate scheduler milestones found for MSD and MSD+REx. We retain the model with best validation Ensemble (seen) accuracy for our final results presented in Sec. 4. As performance of the best checkpoints of MSD with milestones 50 and 100, and respectively MSD+REx with milestones 97 and 100, are very close, in both cases we evaluated the final models on the test set and kept the best in each case (MSD with milestone 50 and MSD+REx with milestone 97), observing only minor differences between each defense's pair of choices.

C.2. Avg model

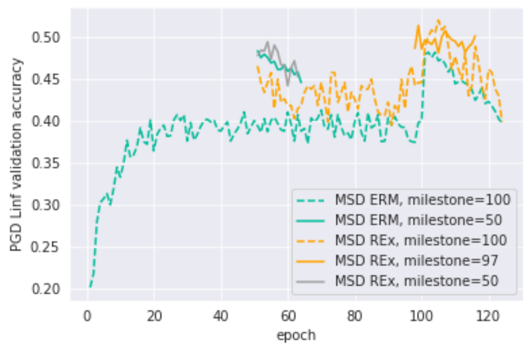
As argued in our introduction, it is somewhat surprising that REx successfully improved MSD due to MSD being a single-domain baseline. REx was originally designed to be used with baselines where multiple domains appear in the loss, and in particular ERM over multiple domains. Fig. 8 illustrates how REx clearly benefits the Avg baseline more, with very little tuning effort required to achieve results above all baselines as reported in subsection 4.3.



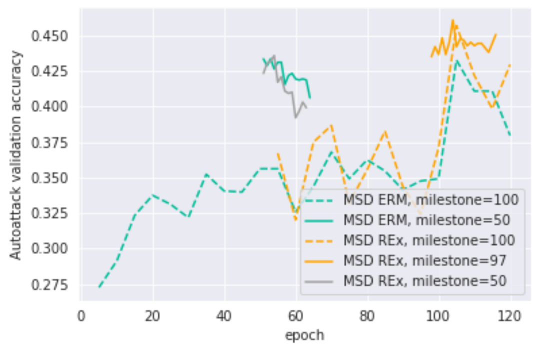
(a) P_1 accuracy



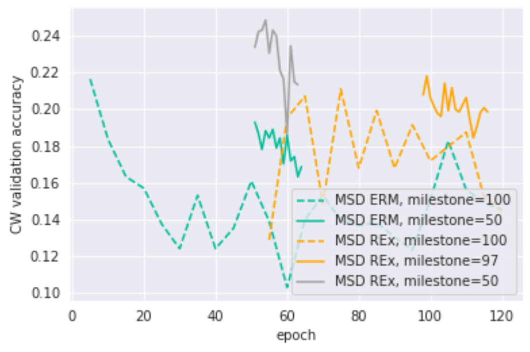
(b) P_2 accuracy



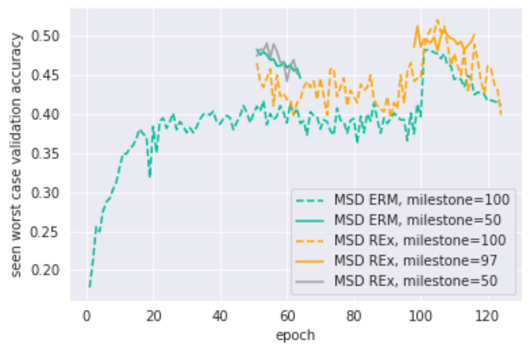
(c) P_∞ accuracy



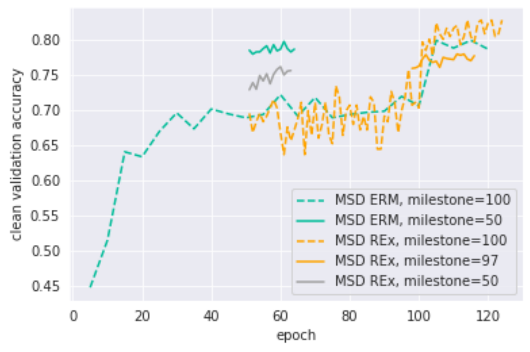
(d) AutoAttack$_\infty$ accuracy



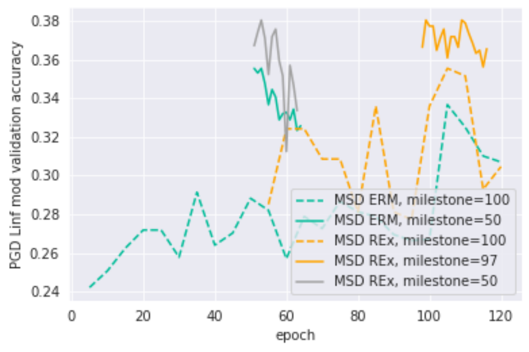
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy

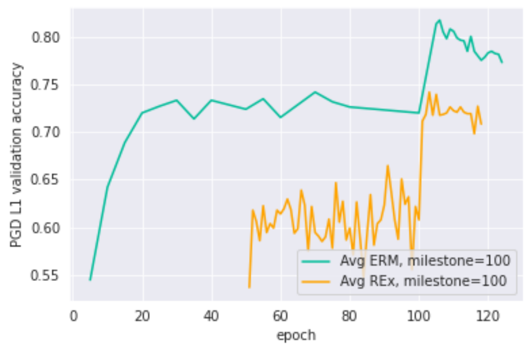


(g) No attack (clean) accuracy

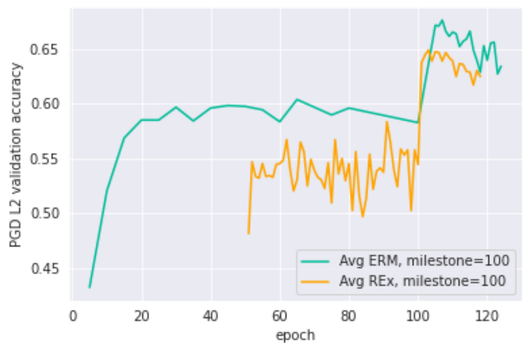


(h) $P_\infty^\bullet$ accuracy

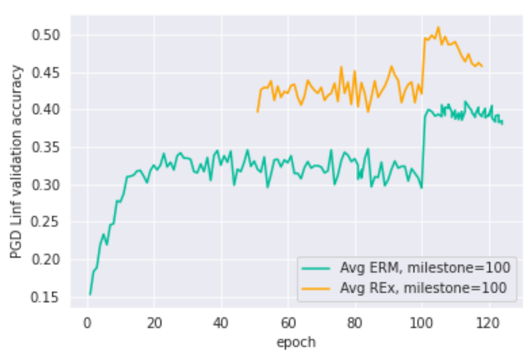
Figure 7: Validation accuracy of MSD and MSD+REx on CIFAR10 on various attacks with different milestones for the learning rate decay. Early stopping is performed for each model at the peak of the ensemble (seen) accuracy. The MSD model with a milestone at epoch 50 and the MSD+REx model with a milestone at epoch 97 are the final models retained in subsection 4.3.



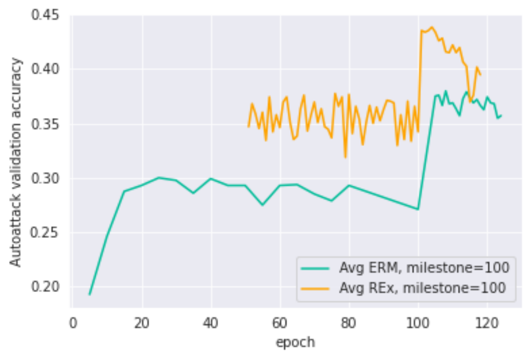
(a) P_1 accuracy



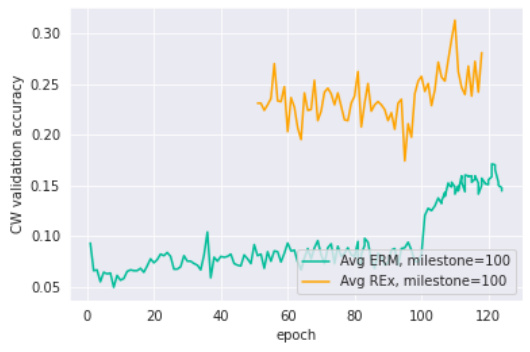
(b) P_2 accuracy



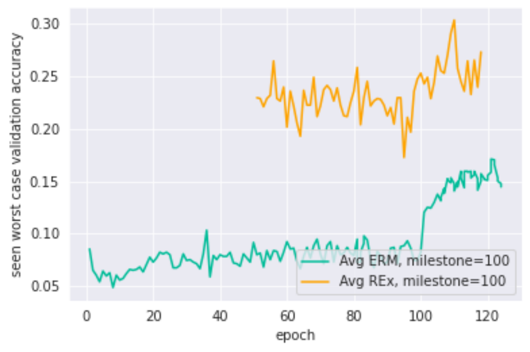
(c) P_∞ accuracy



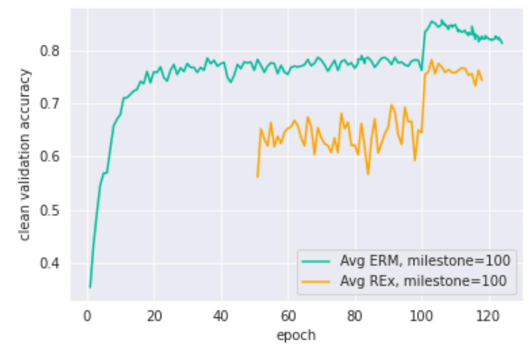
(d) AutoAttack$_\infty$ accuracy



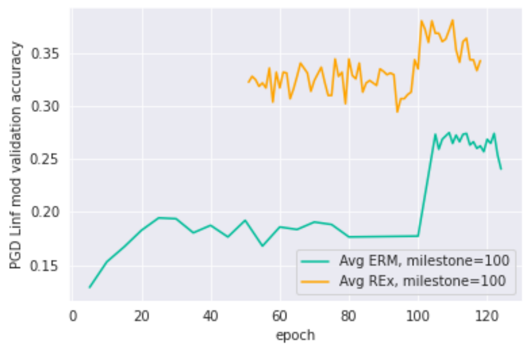
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy



(g) No attack (clean) accuracy



(h) $P_\infty^\bullet$ accuracy

Figure 8: Validation accuracy of Avg and Avg+REx on CIFAR10 on various attacks with learning rate decay milestone at epoch 100. This illustrates how much easier it is to get improvements with REx on baselines based on ERM over multiple domains.

D. Effect of varying REx's β coefficient

In Fig. 9 we investigate the effect of varying β in REx+MSD. Note that the hyperparameters of both MSD and MSD+REx are suboptimally tuned in this figure, which only aims to illustrate the effect of varying β. To generate those figures, both baselines use weight decay, MSD's learning rate milestone is set at epoch 100. MSD+REx loads an MSD checkpoint at epoch 105 with a learning rate set to 0.1 which is decayed at epoch 155.

We observe that while there is some robustness to the choice of β for some domains, the difference is especially large on CW_2 and $P_\infty^\bullet$, where a larger value benefits the model. β also has a fairly large impact on the clean, P_1 and P_2 accuracies. This is explained by the fact that low values of β imply that the variance term will have lower impact and the model will value high performing seen domains (clean, P_1, P_2) more when updating weights than if larger values of β were used. In contrast, large values of β emphasise the variance regularisation which benefits accuracy against stronger attacks more.

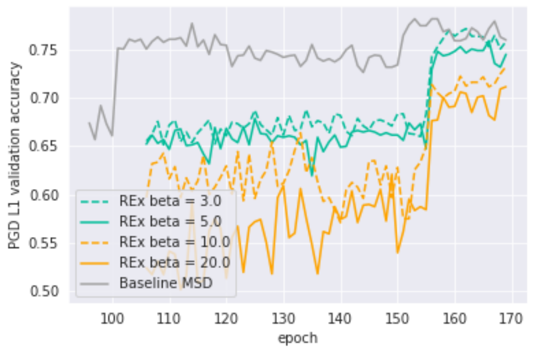
E. Miscellaneous results

- In preliminary experiments without hyperparameter tuning, REx did not benefit a model pretrained solely on P_∞.
- In preliminary experiments without hyperparameter tuning, REx (and the Avg baselines) incurred significant losses in robustness when attempting to train on only one type of perturbation per sample in the batch (in the sense that each sample from a minibatch would only contribute to a single random domain among the seen domains, instead of all seen domains for each sample).
- We attempted to just add the variance penalty term to the MSD baseline while still training on the MSD attack. More explicitly, the ERM term consisted in the loss on the MSD attack, and the variance term separately computed P_1, P_2, P_∞ perturbations. This leads to iterations that are twice as expensive, for no observed benefit. Therefore, we instead prefered to define MSD+REx as performing REx+Avg$_{PGDs}$ on a model pretrained with MSD.
- On CIFAR10, training on $\{P_\infty, DF_\infty, CW_2\}$ is about 8 times more computationally expensive than training on *PGDs* or MSD with 10 iterations per P_p attack (factoring in that in all cases, we validate on all domains every 5 epochs). Since the former leads to a significant advantage in robustness over the ensembles of attacks evaluated here, there is a strong trade-off between computational cost and adversarial robustness when training on those attacks.

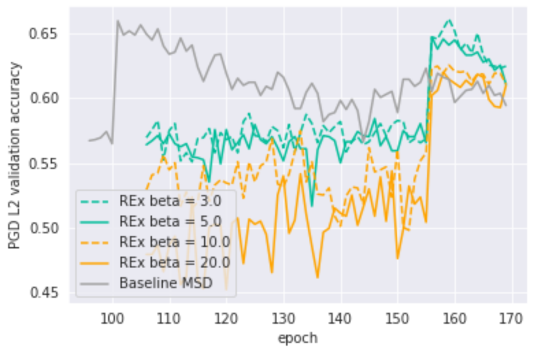
In figures 10 and 11, we show the domains generated for the Avg and Avg+REx models from different attacks, along with the unperturbed data. Above each image is the class predicted by the model, and in parentheses the true class. The classes match the following numbers:

- airplane : 0
- automobile : 1
- bird : 2
- cat : 3
- deer : 4
- dog : 5
- frog : 6
- horse : 7
- ship : 8
- truck : 9

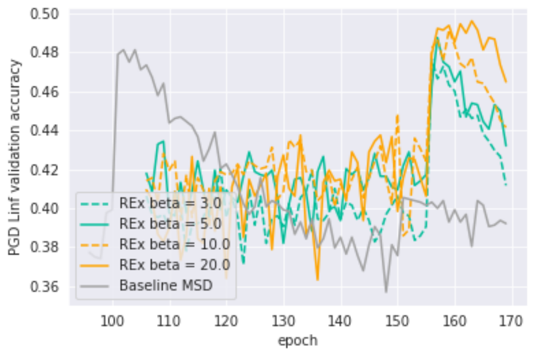
This highlights how in general, these adversarial examples are not difficult to classify for humans. **This however also illustrates how CW_2 perturbations, in spite of being unbounded, tend to be much less perceptible than those stemming from most commonly used bounded attacks, such as P_∞ or AA_∞, at tunings where they have the highest attack success rate by far among attacks considered.**



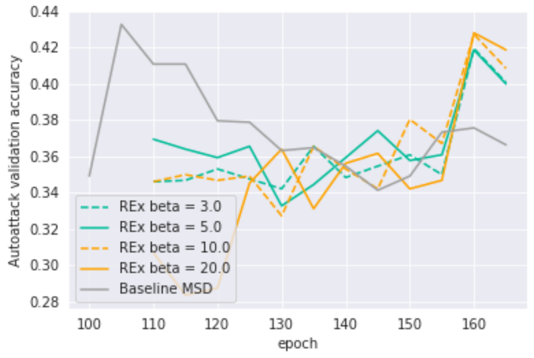
(a) P_1 accuracy



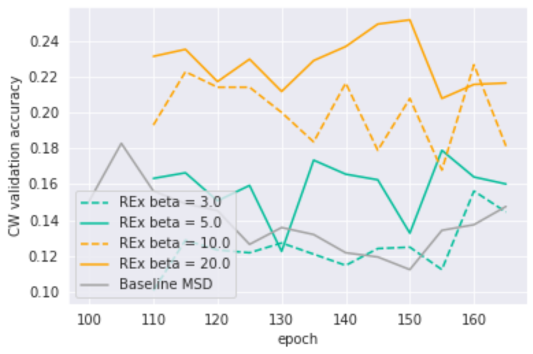
(b) P_2 accuracy



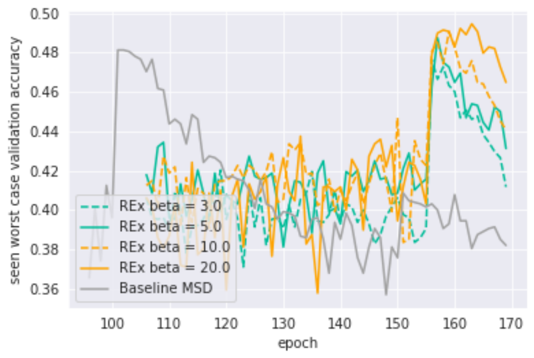
(c) P_∞ accuracy



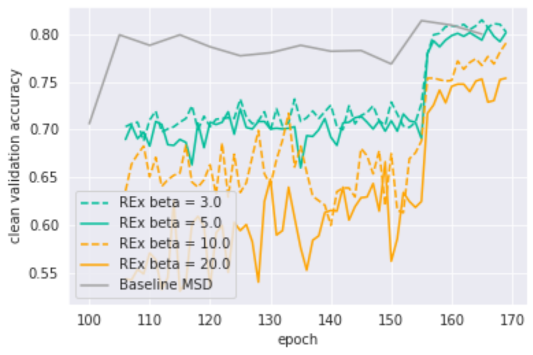
(d) AutoAttack$_\infty$ accuracy



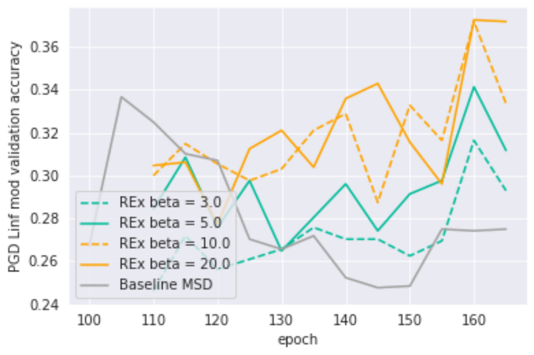
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy

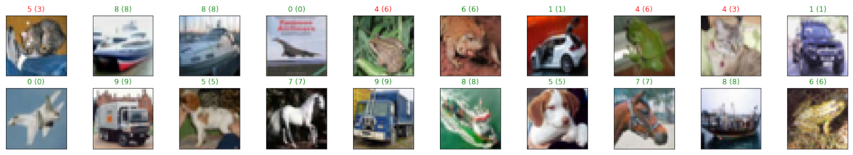


(g) No attack (clean) accuracy

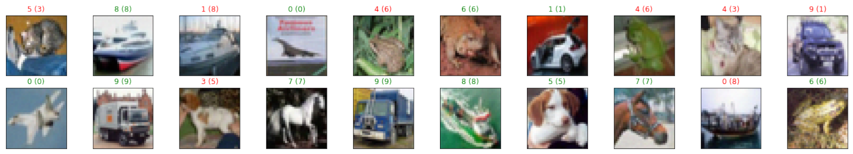


(h) $P_\infty^\bullet$ accuracy

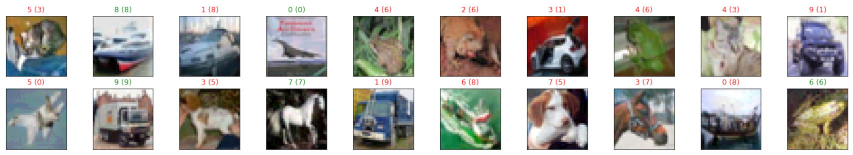
Figure 9: Validation accuracy of MSD and MSD+REx on CIFAR10 on various attacks with different values of β. Note that MSD has a learning rate decay at epoch 100, and MSD+REx at epoch 155.



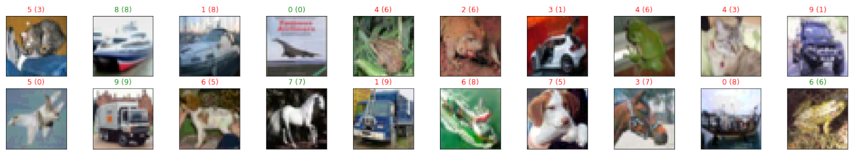
(a) P_1 adversarial examples



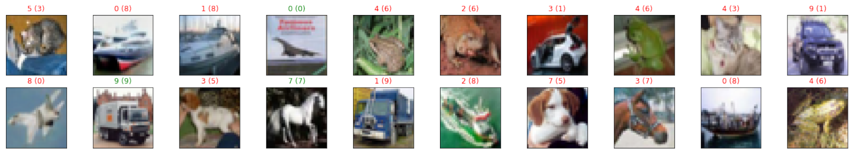
(b) P_2 adversarial examples



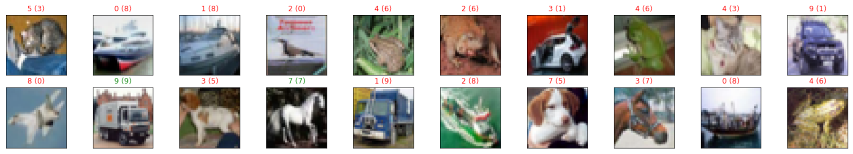
(c) P_∞ adversarial examples



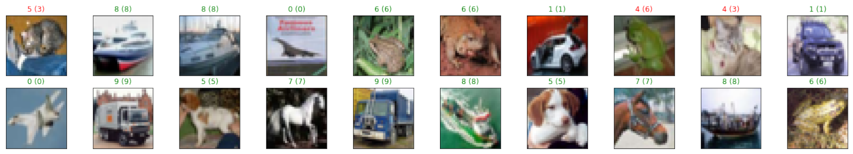
(d) AutoAttack$_\infty$ adversarial examples



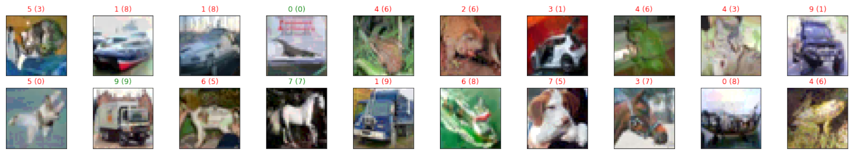
(e) CW$_2$ adversarial examples



(f) CW$_2^\bullet$ examples

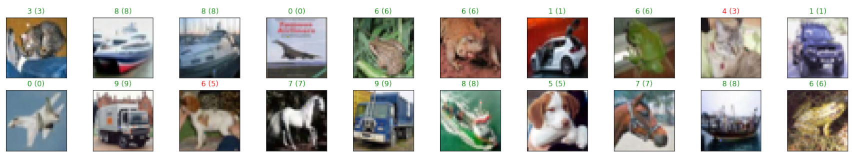


(g) No attack (clean) adversarial examples

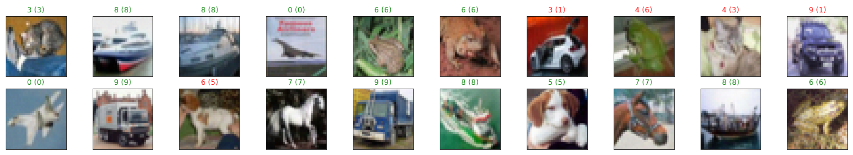


(h) $P_\infty^\bullet$ adversarial examples

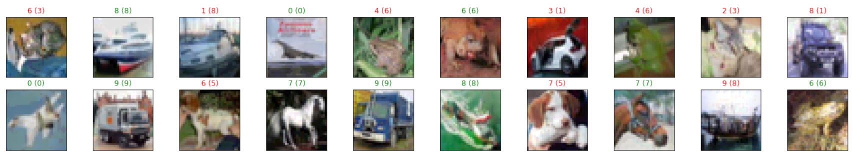
Figure 10: Adversarial examples generated from the hyperparameter-optimised Avg model, for each attack. The predicted class and in parentheses, the true class, are displayed above each image.



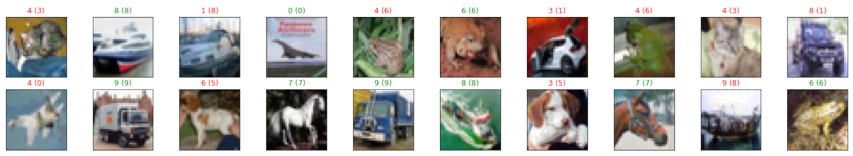
(a) P_1 adversarial examples



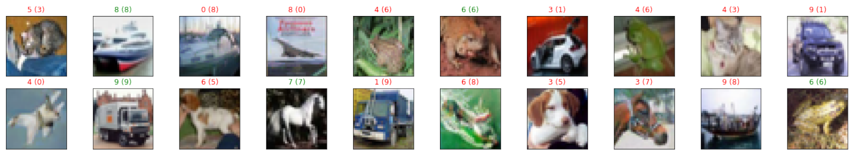
(b) P_2 adversarial examples



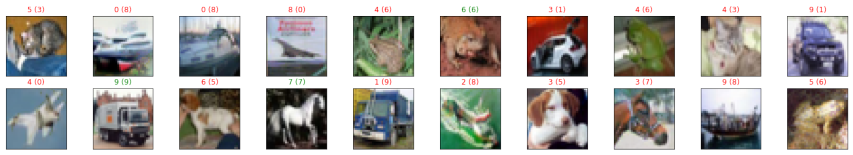
(c) P_∞ adversarial examples



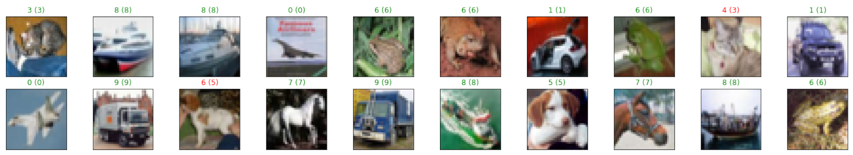
(d) AutoAttack$_\infty$ adversarial examples



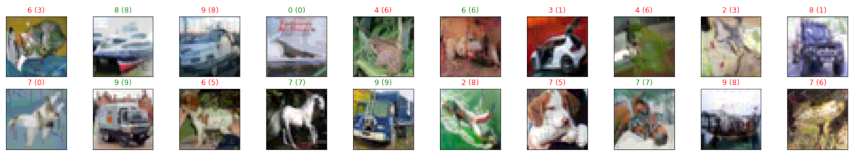
(e) CW$_2$ adversarial examples



(f) CW$_2^\bullet$ adversarial examples



(g) No attack (clean) adversarial examples



(h) $P_\infty^\bullet$ adversarial examples

Figure 11: Adversarial examples generated from the hyperparameter-optimised Avg+REx model, for each attack. The predicted class and in parentheses, the true class, are displayed above each image.