TOWARDS OUT-OF-DISTRIBUTION ADVERSARIAL ROBUSTNESS

Anonymous authors
Paper under double-blind review

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% the best existing baseline to 25.9% on MNIST, and from 16.9% to 23.5% on CIFAR10.

1 INTRODUCTION

Vulnerability to adversarial perturbations (Biggio et al., 2013; Szegedy et al., 2014; Goodfellow et al., 2015) is a major concern for real-world applications of machine learning such as healthcare (Qayyum et al., 2020) and autonomous driving (Deng et al., 2020). For example, Eykholt et al. (2018) show how seemingly minor physical modifications to road signs may lead an autonomous car into misinterpreting stop signs, while Li et al. (2020) achieve high success rates with over-the-air adversarial attacks on speaker systems.

Much work has been done in defending against adversarial attacks (Goodfellow et al., 2015; Papernot et al., 2016). However, 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 improve robustness by training on Projected Gradient Descent (PGD) 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.

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 the proxy for robustness, a better notion of robustness, 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, in the example of autonomous driving, an attacker will not be constrained to a single attack on stop signs, and is free to 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.

There are multiple challenges: first, Gulrajani & Lopez-Paz (2020) show that domain generalisation methods, such as REx, often fail to improve over empirical risk minimisation in many settings. Second, these methods are usually designed for stationary settings, whereas in adversarial machine learning, the distribution of adversarial perturbations is non-stationary during training as the attacks adapt to the changes in the model parameters. Finally, the multiperturbation defense proposed by Maini et al. (2020) does not use multiple domains, which REx originally requires.

Therefore, we are interested in the two following research questions:

1. Can REx improve the robustness against multiple attacks seen during training?
2. Can REx improve the robustness against unseen attacks, that is, attacks seen only 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, hyperparameter 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 (i.e. adversarial attacks) have been proposed in the literature (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.

One of the earliest attacks, the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), computes a perturbation on an input x^0 by performing a step of sign gradient ascent in the direction that increases the loss L the most, given the model's current parameters θ. This yields an adversarial example $\tilde{x}$ that may be misclassified:

$$\tilde{x} = x^0 + \alpha \operatorname{sgn}(\nabla_x L(\theta, x^0, y)). \tag{1}$$

This was later enhanced into the Projected Gradient Descent (PGD) attack (Kurakin et al., 2017; Madry et al., 2018) by iterating multiple times this operation and adding projections to constrain it to some neighbourhood of x^0, usually a ball of radius ϵ centered at x^0, noted $\mathcal{B}_\epsilon(x^0)$:

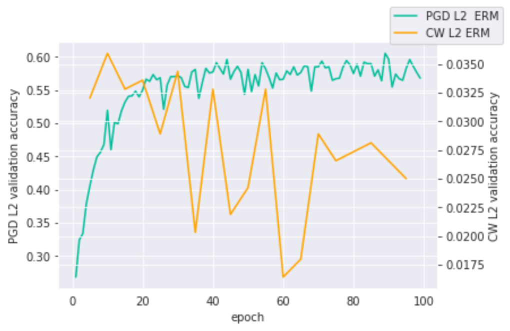
$$x^{t+1} = \Pi_{\mathcal{B}_\epsilon(x^0)} \left(x^t + \alpha \operatorname{sgn}(\nabla_x L(\theta, x^t, y)) \right). \tag{2}$$

With the advent of diverse algorithms to *defend* classifiers against such attacks, approaches for discovering adversarial examples have become increasingly more complex over the years. Notably, it was discovered that a great number of adversarial defenses rely on *gradient obfuscation* (Athalye et al., 2018), which consists in learning how to mask or distort the classifier's gradients to prevent attacks iterating over gradients from making progress. However, it was later discovered that such approaches can be broken by other attacks (Athalye et al., 2018; Croce & Hein, 2020), some of which simply circumvent these defenses by not relying on gradients (Brendel et al., 2019; Andriushchenko et al., 2020).

A defense that was shown to be robust to such countermeasures is Adversarial Training (Madry et al., 2018), which usually consists in training on PGD adversarial examples. Adversarial training corresponds to solving a minimax optimisation problem where the inner loop executes an adversarial attack algorithm, usually PGD, to find pertubations to the inputs that maximise the classification loss, while the outer loop tunes the network parameters to minimise the loss on the adversarial examples. Despite the method's simplicity, robust classifiers trained with adversarial training achieve state-of-the-art levels of robustness against various newer attacks (Athalye et al., 2018; Croce & Hein, 2020). For this reason, adversarial training has become one of the most common methods for training adversarially robust neural networks.

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. Another reason is that different attacks, such as PGD and the Carlini and Wagner (Carlini & Wagner, 2017) attacks, optimise different losses (note that this is also true for PGD of different norms). As an example, Fig. 1 illustrates how, when training adversarially a model on PGD with respect to the L_2 norm, the accuracy against one attack may improve while it may decrease against another attack, even if the attacks use the same p-norm.

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



Highlighting the need for methods specific to defending against multiple types 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, and the max strategy, which trains on the attack with the highest loss for each sample:

$$L_{\text{Avg}}(\theta, \mathcal{A}) = \mathbb{E}\frac{1}{|\mathcal{A}|}\sum_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{3}$$

$$L_{\max}(\theta, \mathcal{A}) = \mathbb{E}\max_{A\in\mathcal{A}} \ell(\theta, A(x), y) \tag{4}$$

Maini et al. (2020) propose a modification to the max method of Tramèr & Boneh (2019): instead of having 3 different PGD adversaries that each iterate over a budget of iterations as in eq. 2, they design an attack consisting in choosing the worst perturbation among 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*[1] *only consists in training on a single attack.* Maini et al. (2020) show that, in their experimental setup, MSD yields superior performance to both the Avg and max approaches of Tramèr & Boneh (2019).

Nevertheless, there is still a very 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 that of 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

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

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.

Our work views adversarial robustness as a domain generalisation problem. In our case, the domains correspond to 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. It is natural to frame adversarial robustness as a domain generalisation problem, because 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 on 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 of Tramèr & Boneh (2019) can be interpreted as doing domain generalisation with ERM over the 3 PGD adversaries as training domains. Similarly, the max approach consists in applying the 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 on different domains. This encourages worst-case robustness over more extreme versions of the shifts (in this case, shifts are between different attacks) observed between the training domains. This can be motivated in the setting of adversarial robustness by the observation that adversaries might shift 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:

$$\mathcal{R} = 1 - \mathbb{E} \max_{D \in \mathcal{D}} \ell_{01}(\theta, D(x), y) \tag{5}$$

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:

$$L_{\text{REx}}(\theta, \mathcal{D}) = L_{\text{Avg}}(\theta, \mathcal{D}) + \beta \operatorname*{Var}_{D \in \mathcal{D}} \mathbb{E}\, \ell(\theta, D(x), y) \tag{6}$$

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 use batch sizes of 128 when training on both datasets.

Optimiser – We use Stochastic Gradient Descent (SGD) with a momentum of 0.9. In subsections 4.1 and 4.2 we do not perform hyperparameter optimisation to isolate the effect of REx from interactions with hyperparameter tuning, which would differ for each individual 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 4.4, we perform hyperparameter optimisation. 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 (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 $\{P_\infty^\bullet, DF_\infty^\bullet, CW_2^\bullet, \mathit{AutoAttack}_\infty\}$, with additionally $\mathit{AutoAttack}_2$ in subsection 4.4. 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 restarts for each attack per sample to reduce randomness in the evaluations (but not during training).

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.

All models are trained using a single Nvidia A100 for MNIST and 2 Nvidia A100s for CIFAR10.

4 Results

In this section, we present our results without performing individual hyperparameter optimisation in subsections 4.1 and 4.2, and after tuning the hyperparameters of each defense in subsection 4.4. We find that REx consistently improves the baselines over ensembles of seen or unseen attacks. REx tends to most often sacrifice a little performance on the best performing domains (usually, the unperturbed data, P_1 and P_2) but improves the worst-case accuracy by a much larger amount. This is the case even when using REx on MSD. In general, we find that REx is relatively robust to the choice of β, albeit it can be used to tune relative performance between easier and harder domains (see Appendix C). More implementation details and observations can generally be found in the appendix.

4.1 MNIST

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.

	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_∞	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

We report our results on MNIST in Table 1. REx significantly improves the 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 $P_\infty^\bullet$ 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 improve performance significantly, ruling out the absence of schedule as a cause. While we use the original code of Maini et al. (2020), this might be because we did not use the same architecture as them on MNIST. Moreover, 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 is failing to learn how to be robust against P_∞ and AutoAttack. 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 multiperturbation baselines.

Key observation 1 (no hyperparameter tuning): REx improve the 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 attacks, by sacrificing a little robustness against the weakest individual attacks.

4.2 CIFAR10

The results on CIFAR10 are summarised in Table 2. We still observe 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,

Table 2: 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**

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 of 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).

Interestingly, the model adversarially 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.

Key observations 2 (no hyperparameter tuning):

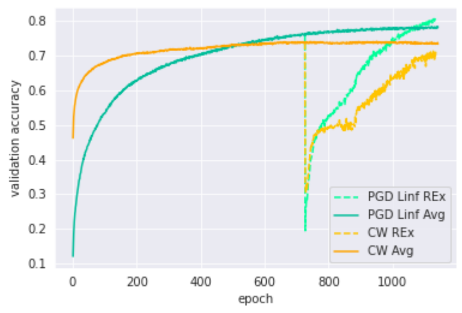
- REx improve the performance of all baselines on CIFAR10 with a ResNet18, from 10.7% with the best baseline to 17.9% accuracy against an ensemble of attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multiperturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when using REx.

4.3 Early stopping and REx

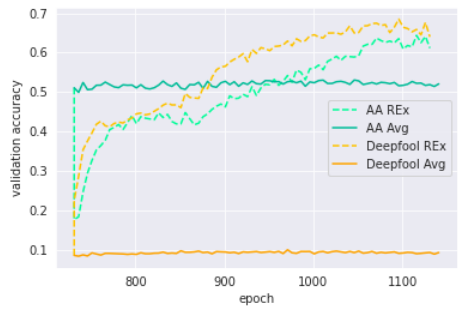
In Figure 2, we observe how regularising by the variance over the domains leads to better peak performance when using REx, over different datasets, architectures, and baselines. Like the baselines, REx requires early stopping. For all defenses, we use the validation set to choose when to early stop, by selecting the epoch when the performance peaks on the ensemble of seen domains. As shown on the MNIST 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 Appendix B for more details on early stopping and more validation curves.

4.4 CIFAR10 with hyperparameter optimisation

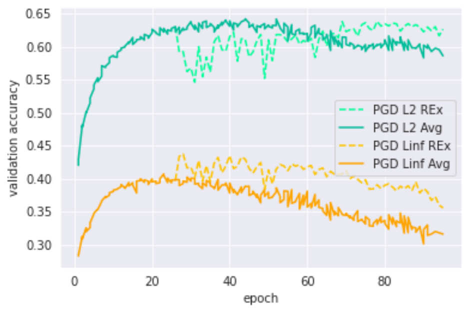
Unlike in previous subsections, we use weight decay and attempt to find an optimal learning rate schedule for each defense. The results are summarised in Table 3. Due to its worse performance compared to the other baselines on CIFAR10 with a ResNet18 (which we confirm to be especially true when tuning hyperparameters in preliminary experiments), we chose not to use the Avg$_{PGDs}$ baseline here.



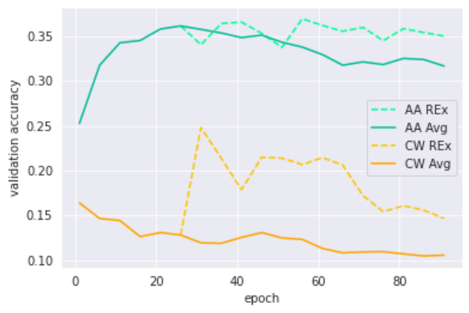
(a) MNIST seen attacks



(b) MNIST unseen attacks



(c) CIFAR10 seen attacks



(d) CIFAR10 unseen attacks

Figure 2: Validation accuracy of Avg on MNIST (top) and MSD on CIFAR10 (bottom) with and without REx (dashed line), against seen attacks (left) and unseen attacks (right). AA denotes AutoAttack.

Table 3: 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**

The cumulative effect of weight decay and learning rate schedules significantly improves all defenses' robustness, as shown by Rice et al. (2020) and Pang et al. (2020). Once again, we observe that REx improves significantly the seen and unseen ensemble accuracies over the baselines. Moreover, with hyperparameter tuning, REx appears to lose less 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 when training on $\{P_\infty, DF_\infty, CW_2\}$. As before, 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 is able to perform better than P_∞ adversarial training on P_∞ attacks.

As suspected, there are interaction effects between hyperparameter tuning and the performance of REx relative to a baseline, in the case of MSD. Hyperparameter tuning of MSD+REx, in the sense of choosing at which epoch to start using REx, and when to decay the learning rate, is sensitive. 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 performing ERM on multiple domains. Therefore, when REx is used on MSD, REx uses the loss indicated in eq. 6. However, because the Avg$_{PGDs}$ baseline performs significantly worse than the MSD baseline, especially with tuned hyperparameters, 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.

Key observations 3 (with hyperparameter tuning):

- REx improve 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 attacks, by sacrificing a little robustness against the weakest individual attacks.
- Multiperturbation defenses only achieve higher P_∞ and worst-case performance than P_∞ adversarial training when they are used in conjunction with REx.

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, the worst-case robustness against the set of available attacks is most reflective of the performance achieved 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 produces consistent performance improvements over baselines across different datasets, architectures, training attacks, test attack types and tunings. One 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 robustness and promising research directions are significant enough to be of interest to the community.

Indeed, our work shows the promise of domain generalisation approaches to adversarial robustness. Future work could investigate other domain generalisation methods, such as Distributionally Robust Optimisation (DRO) (Sagawa et al., 2019) or Invariant Risk Minimisation (IRM) (Arjovsky et al., 2019), and evaluate the effectiveness of these approaches against black-box attackers without access to gradient information.

Our results are particularly interesting in light of the failure of REx to improve performance over well-tuned baselines on non-adversarial domain generalisation benchmarks (Gulrajani & Lopez-Paz, 2020).

Reproducibility statement

We provide the code used for the implementation of all methods and list the hyperparameters used for the models and the attacks. We provide in the appendix details about early stopping epochs, learning rate schedule milestones, etc. We fix and provide the seed in the code, except for adversarial attack computations, where we leave a random seed. In all our experiments, during training, we observe a variance in the order of $\pm 1\%$ in performance for any given defense. The REx implementation is simple and, using the information provided in the main body and appendix, we expect it to be easy to replicate our results. The only experiments that required extensive hyperparameter search were MSD and REx+MSD with tuned hyperparameters. Indeed, as explained, even after finding the best possible hyperparameters for the MSD baseline, it took several attempts to find a learning rate decay milestone for REx+MSD after turning on REx, that provided significant improvements on more than just the Carlini & Wagner attacks. While this is not necessarily surprising when discussing hyperparameter search, we note that in every other experiment, the advantage in using REx did not require several attempts (see Appendix B for a illustration of this). Our strategy for hyperparameter search to solve this issue is explained in Appendix B.

References

Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: a query-efficient black-box adversarial attack via random search. In *European Conference on Computer Vision*, pp. 484–501. Springer, 2020. URL https://arxiv.org/abs/1912.00049.

Martin Arjovsky, Léon Bottou, Ishaan Gulrajani, and David Lopez-Paz. Invariant risk minimization. *arXiv:1907.02893*, 2019.

Anish Athalye, Nicholas Carlini, and David Wagner. 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.

Pouya Bashivan, Reza Bayat, Adam Ibrahim, Kartik Ahuja, Mojtaba Faramarzi, Touraj Laleh, Blake Richards, and Irina Rish. Adversarial feature desensitization. *Advances in Neural Information Processing Systems*, 34, 2021. URL https://arxiv.org/abs/2006.04621.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013. URL https://arxiv.org/abs/1708.06131.

Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. *Advances in neural information processing systems*, 32, 2019. URL https://arxiv.org/abs/1907.01003.

Nicholas Carlini and David Wagner. 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.

Francesco Croce and Matthias Hein. 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.

Yao Deng, Xi Zheng, Tianyi Zhang, Chen Chen, Guannan Lou, and Miryung Kim. An analysis of adversarial attacks and defenses on autonomous driving models. In *2020 IEEE international conference on pervasive computing and communications (PerCom)*, pp. 1–10. IEEE, 2020. URL https://arxiv.org/abs/2002.02175.

Gavin Weiguang Ding, Luyu Wang, and Xiaomeng Jin. Advertorch v0. 1: An adversarial robustness toolbox based on pytorch. *arXiv preprint arXiv:1902.07623*, 2019. URL https://arxiv.org/abs/1902.07623.

Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1625–1634, 2018.

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

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 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.

Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. 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.

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

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

David Krueger, Ethan Caballero, Joern-Henrik Jacobsen, Amy Zhang, Jonathan Binas, Dinghuai Zhang, Remi Le Priol, and Aaron Courville. 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.

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

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

Zhuohang Li, Cong Shi, Yi Xie, Jian Liu, Bo Yuan, and Yingying Chen. Practical adversarial attacks against speaker recognition systems. In *Proceedings of the 21st international workshop on mobile computing systems and applications*, pp. 9–14, 2020. URL https://par.nsf.gov/servlets/purl/10193609.

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

Pratyush Maini, Eric Wong, and Zico Kolter. 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.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 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.

Tianyu Pang, Xiao Yang, Yinpeng Dong, Hang Su, and Jun Zhu. Bag of tricks for adversarial training. In *International Conference on Learning Representations*, 2020. URL https://arxiv.org/abs/2010.00467.pdf.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, pp. 582–597. IEEE, 2016. URL https://arxiv.org/abs/1511.04508.

Adnan Qayyum, Junaid Qadir, Muhammad Bilal, and Ala Al-Fuqaha. Secure and robust machine learning for healthcare: A survey. *IEEE Reviews in Biomedical Engineering*, 14:156–180, 2020. URL https://arxiv.org/abs/2001.08103.

Leslie Rice, Eric Wong, and Zico Kolter. 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.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. 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.

Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Improving the generalization of adversarial training with domain adaptation. *arXiv preprint arXiv:1810.00740*, 2018. URL https://arxiv.org/abs/1810.00740.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. URL http://arxiv.org/abs/1312.6199.

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

Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Wenjun Zeng, and Tao Qin. Generalizing to unseen domains: A survey on domain generalization. *arXiv preprint arXiv:2103.03097*, 2021. URL https://arxiv.org/abs/2103.03097.

The code can be found at [redacted during review].

A More on methodology

A.1 Attack tunings

Using Advertorch's and Croce's implementation of AutoAttack's[2] parameter names, we report the attacks' tuning here.

MNIST

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.

CIFAR10

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.

CIFAR10 with hyperparameter optimisation
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.

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

A.2 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.

A.2.1 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. 2 where we stopped training for example at epoch 1125 for both the Avg and the Avg+REx models. For CIFAR10, 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. B 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. 2 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.3 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.

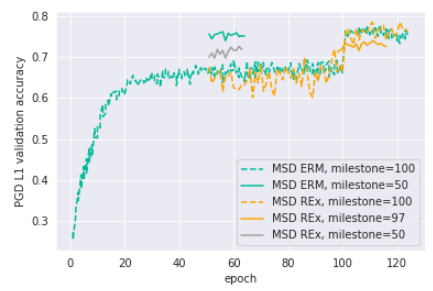
A.4 Other implementation details

We use the implementation of `https://github.com/kuangliu/pytorch-cifar/blob/master/models/resnet.py` for ResNet18.

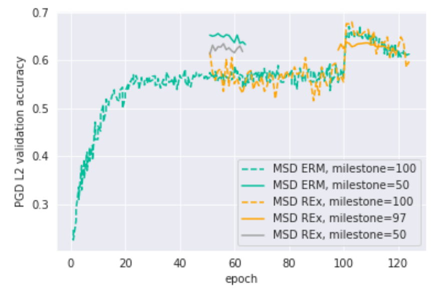
REx's β parameter is generally set to 10, except for MSD+REx on MNIST in subsection 4.1 where it is set to 4. These numbers initially come from setting β to a value of the same order of magnitude as $\frac{L_{Avg}}{\mathrm{Var}}$'s value at the epoch REx is activated, in the early iterations of our experiments. 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 $\frac{L_{Avg}}{\mathrm{Var}} \simeq 30$. See Sec. C for a discussion of how the choice of β affects performance.

B When to early stop, and how milestones affect performance

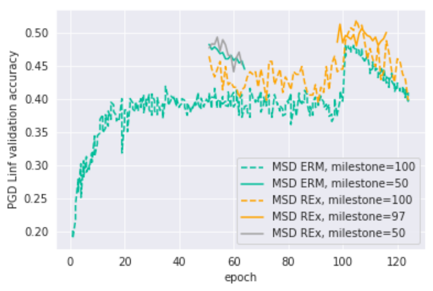
B.1 MSD model



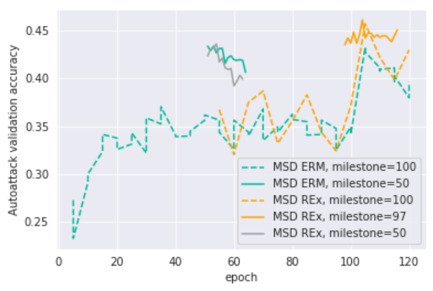
(a) P_1 accuracy



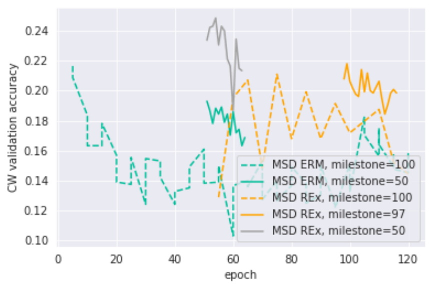
(b) P_2 accuracy



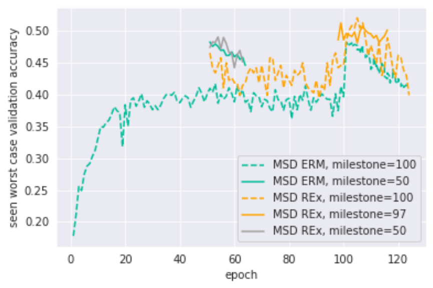
(c) P_∞ accuracy



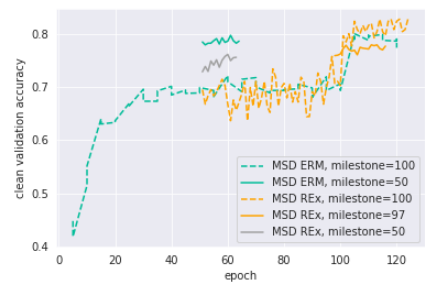
(d) AutoAttack$_\infty$ accuracy



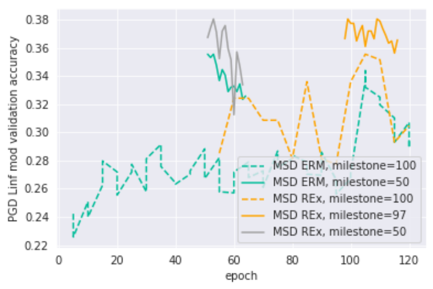
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy



(g) No attack (clean) accuracy



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

Figure 3: 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 4.4.

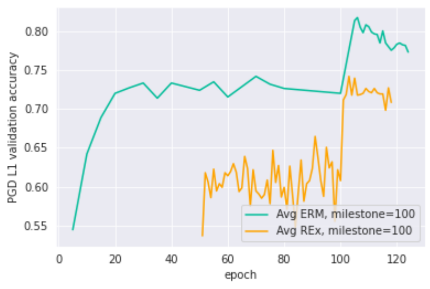
In this subsection, we illustrate two points using Fig. 3: how early stopping is performed on the MSD and MSD+REx models, and how the choice of learning rate decay milestone affects performance. Regarding the former, we early stop based on the peak of the validation Ensemble (seen) accuracy. We motivated 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.

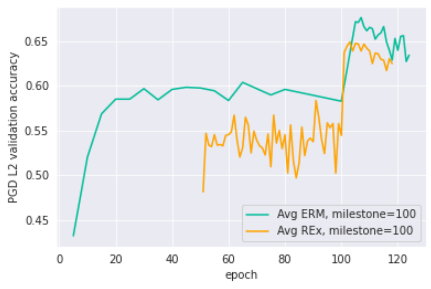
Regarding how milestone choice affects performance, this illustrates how we searched for hyperparameters. We attempt learning rate decays milestones at various epochs, which we then evaluate only for a few epochs, as the performance decays fast anyway as predicted by Rice et al. (2020). The curves in Fig. 3 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.

B.2 AVG MODEL

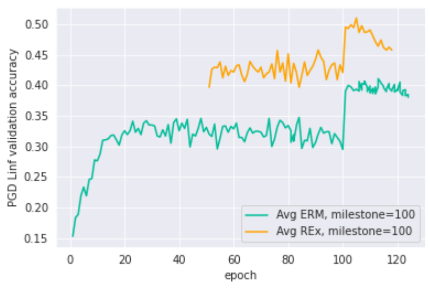
As argued in our introduction, it is somewhat surprising that REx successfully improved MSD due to MSD being a single-domain baseline. REx was originally designed to be used with baselines where multiple domains appear in the loss, and in particular ERM over multiple domains. Fig. 4 illustrates how REx clearly benefits the Avg baseline more, with very little tuning effort required to achieve results above all baselines as reported in subsection 4.4.



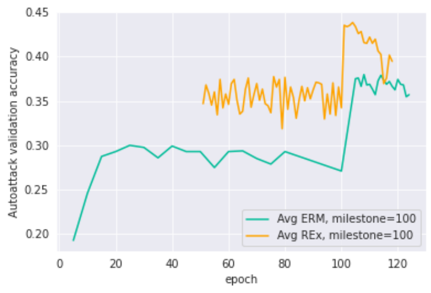
(a) P_1 accuracy



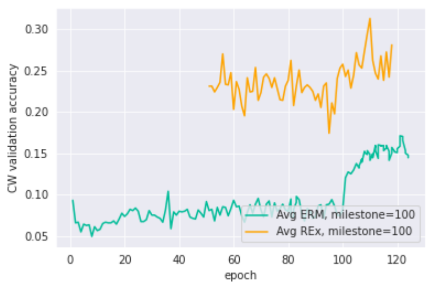
(b) P_2 accuracy



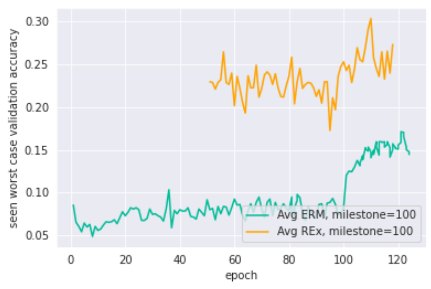
(c) P_∞ accuracy



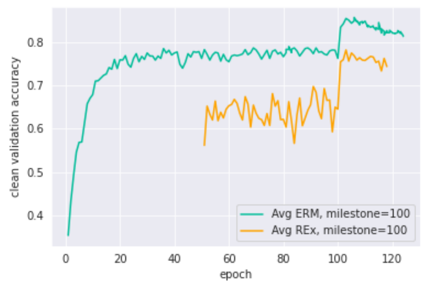
(d) AutoAttack$_\infty$ accuracy



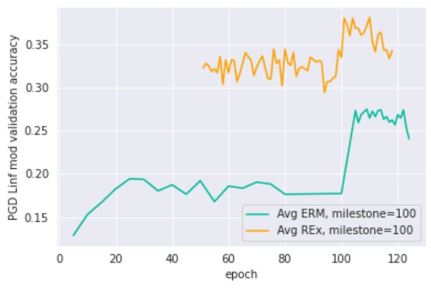
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy



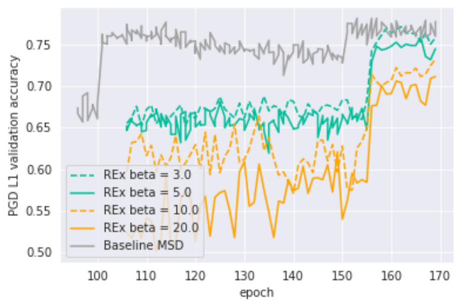
(g) No attack (clean) accuracy



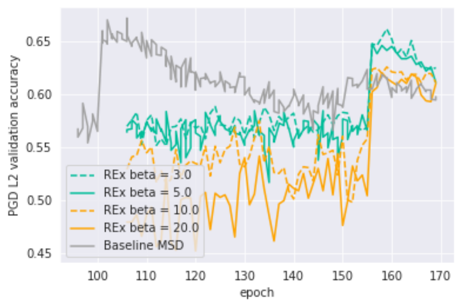
(h) $P_\infty^\bullet$ accuracy

Figure 4: 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.

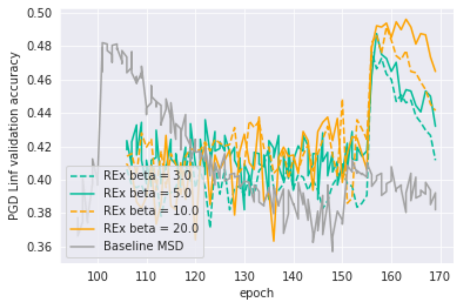
C Effect of varying REx's β coefficient



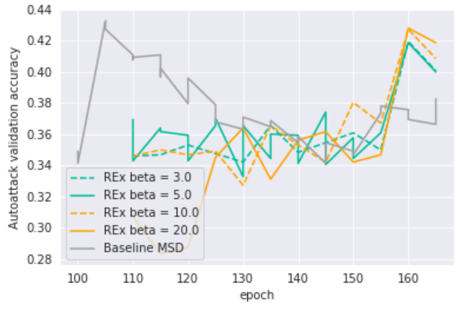
(a) P_1 accuracy



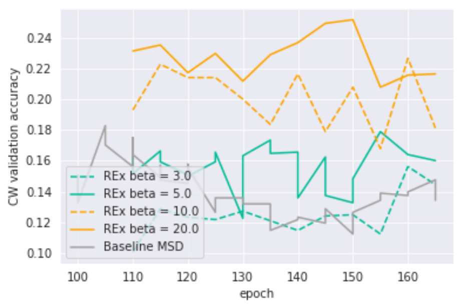
(b) P_2 accuracy



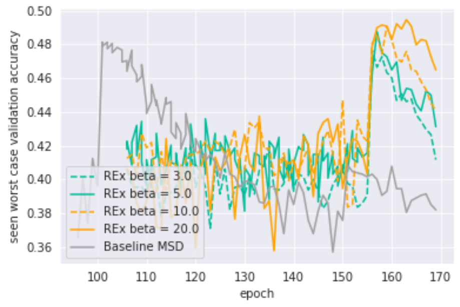
(c) P_∞ accuracy



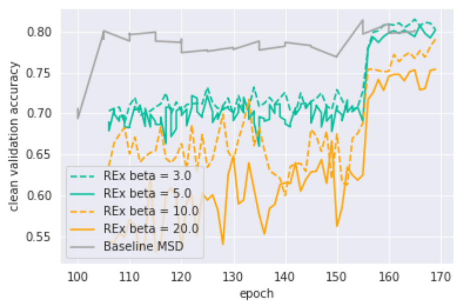
(d) AutoAttack$_\infty$ accuracy



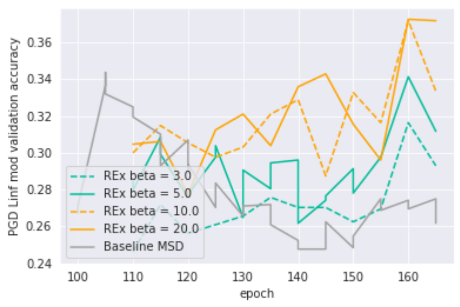
(e) CW$_2$ accuracy



(f) Ensemble (seen) accuracy



(g) No attack (clean) accuracy



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

Figure 5: 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.

In Fig. 5 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.

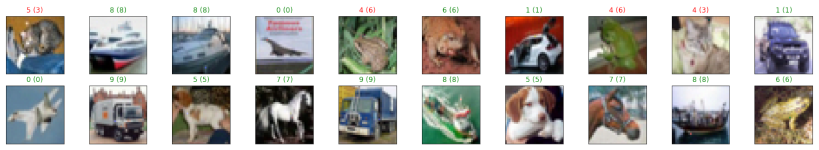
D Miscellaneous results

- In preliminary experiments without hyperparameter tuning, REx did not benefit a model trained solely on P_∞.
- 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.

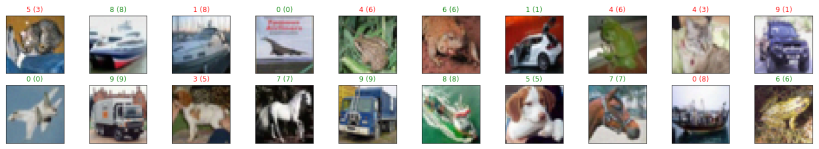
In figures 6 and 7, we show the domains generated for the Avg and Avg+REx models from different attacks, along with the unperturbed data. Above each image is the class predicted by the model, and in parentheses the true class. The classes match the following numbers:

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.

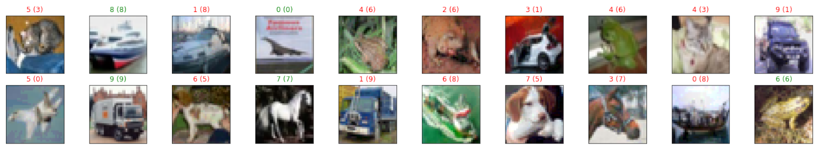


(a) P_1 adversarial examples

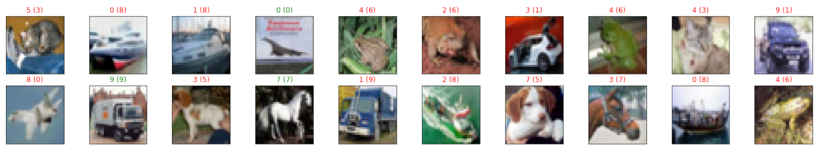


(b) P_2 adversarial examples

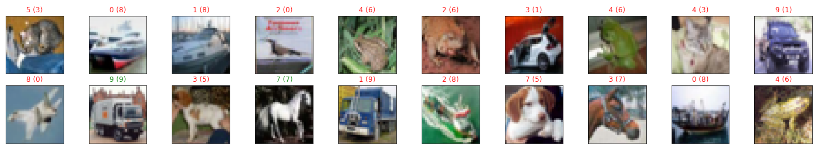
(c) P_∞ adversarial examples



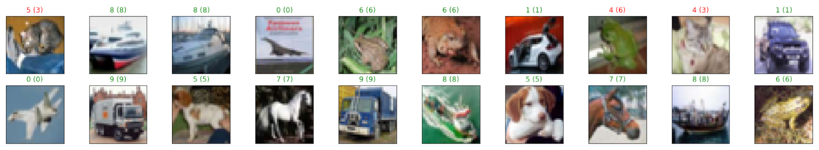
(d) AutoAttack$_\infty$ adversarial examples



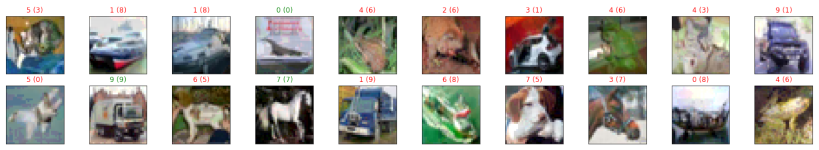
(e) CW$_2$ adversarial examples



(f) CW$_2^\bullet$ examples

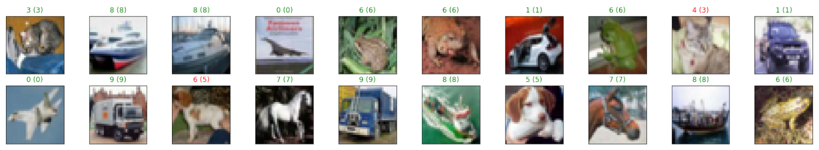


(g) No attack (clean) adversarial examples

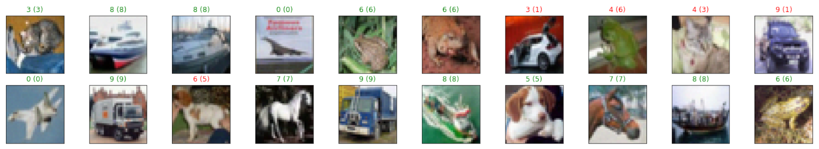


(h) $P_\infty^\bullet$ adversarial examples

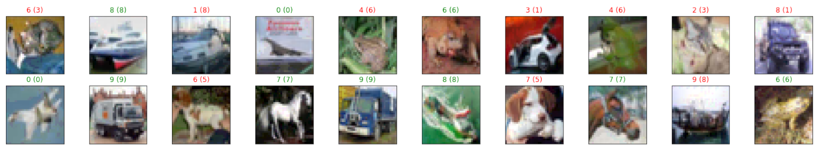
Figure 6: Adversarial examples generated from the hyperparameter-optimised Avg model, for each attack.



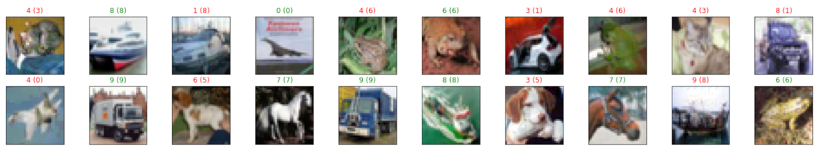
(a) P_1 adversarial examples



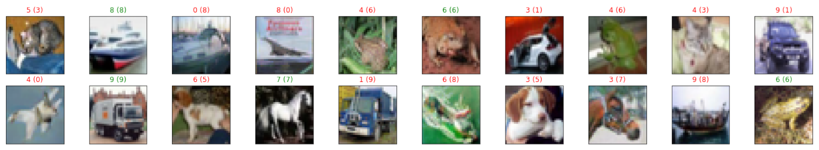
(b) P_2 adversarial examples



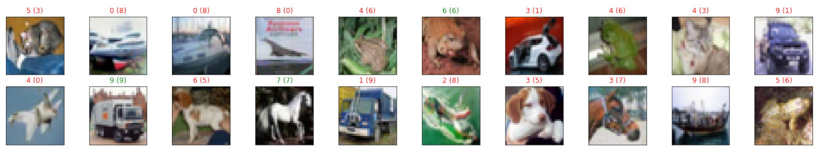
(c) P_∞ adversarial examples



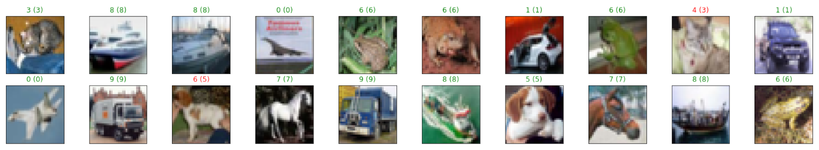
(d) AutoAttack$_\infty$ adversarial examples



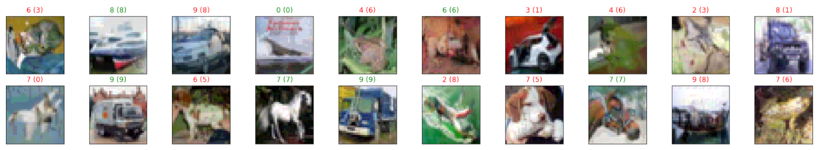
(e) CW$_2$ adversarial examples



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



(g) No attack (clean) adversarial examples



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

Figure 7: Adversarial examples generated from the hyperparameter-optimised Avg+REx model, for each attack.