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 space for improvement against many commonly used attacks by adopting a domain generalisation approach. In particular, we treat different attacks as domains, and apply the method of Risk Extrapolation (REx), which encourages similar levels of robustness against all training attacks. Compared to existing methods, we obtain similar or superior adversarial robustness on attacks seen during training. More significantly, we achieve superior performance on families or tunings of attacks only encountered at test time. On ensembles of attacks, this improves the accuracy from 3.4% on the best existing baseline to 25.9% on MNIST, and from 10.7% to 17.9% on CIFAR10.

1. Intro

In the recent years, deep learning has encountered massive success in many applications, ranging from computer vision to natural language processing. A major concern for several real-world applications of machine learning, such as healthcare (Qayyum et al., 2020) or autonomous driving (Deng et al., 2020), is their vulnerability to adversarial perturbations (Biggio et al., 2013; Szegedy et al., 2014; Goodfellow et al., 2015). 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.

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

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

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 to train directly on the training data perturbed with the Fast Gradient Sign Method, and Madry et al. (2018) further improved robustness by training on an iterative version of this attack called Projected Gradient Descent (PGD), using the L_∞ norm to constrain the search region of adversarial examples.

Unfortunately, adversarial training can fail to provide high robustness against several attacks, or tuning 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, to reuse 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 when designing an adversarial modification to the road sign.

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 different 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 the variance on the different training domains of the empirical risk minimisation loss, as a loss penalty. We choose this method as it is conceptually simple, its iterations are no more costly than existing multi-perturbation baselines', it does not force a choice of 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 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. We show that REx consistently yields benefits across variations in: datasets, architectures, multi-perturbation defenses, 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 examples (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 minuscule perturbation vectors with constrained magnitude that when added to the network's input leads 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 circumventing 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), i.e. training on PGD adversarial examples. Adversarial training consists of solving a minimax optimisation problem where the inner loop executes an adversarial attack algorithm, usually PGD, to find pertubations to the inputs that maximises 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 using 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 have mutually exclusive regions. Another reason is that different attacks, such as PGD and the Carlini and Wagner (Carlini & Wagner, 2017) attacks, optimise different losses.

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_n is PGD with a search region constrained by the L_n norm. The 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:

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

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

Maini et al. (2020) propose a modification to the 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. Maini et al.

(2020) show that, in their experimental setup, MSD[1] 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 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 (Wang et al., 2021). However, recent work demonstrates that such 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.

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.

A key difference with most work in domain generalisation, however, 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. 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 Distributionally 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.

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 (Goodfellow, 2019). 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 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 (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

[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.

Table 1: Accuracy on MNIST for different domains. Highlighted cells indicate that the attack (row) was used during training by the defense (column). Bold numbers indicate an improvement of at least 1% accuracy over the baseline on which REx was pretrained. 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
$P_\infty^\bullet$	0.0	0.0	0.0	5.1	0.6	**4.0**	0.9	0.7	0.2	1.0
$DF_\infty^\bullet$	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 (always unseen)	0.0	0.0	0.0	9.3	3.4	**34.6**	1.2	**8.1**	0.6	**3.9**
Ensemble (all unseen)	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**

some commonly used attacks. On MNIST, we use a 3-layer multilayer 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 during training on both datasets.

Optimiser – We use Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a fixed learning rate of 0.01. While we observe that a reduction of the learning rate benefits all modalities, due to the sheer number of hyperparameters involved, we wish to avoid the added complexity of having to search for an optimal schedule for each method and tuning of attacks. For the same reason, we fix the coefficient β in the REx loss.

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_∞) (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 **always unseen** domains is defined as $\{P_\infty^\bullet, DF_\infty^\bullet, CW_2^\bullet, \textit{AutoAttack}\}$. The set of **all unseen** domains is the set of all domains except those seen during training, and therefore varies between baselines. We perform 10 random 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

4.1. MNIST

We report our results on MNIST in Table 1. REx significantly improves the robustness against the ensembles of

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**
$CW_2^\bullet$	0.0	0.0	0.1	1.1	1.6	**2.6**	1.1	**2.7**	1.0	**2.7**
AutoAttack	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 (always unseen)	0.0	0.0	1.0	20.1	16.3	**24.0**	13.2	**22.0**	19.3	**23.6**
Ensemble (all unseen)	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**

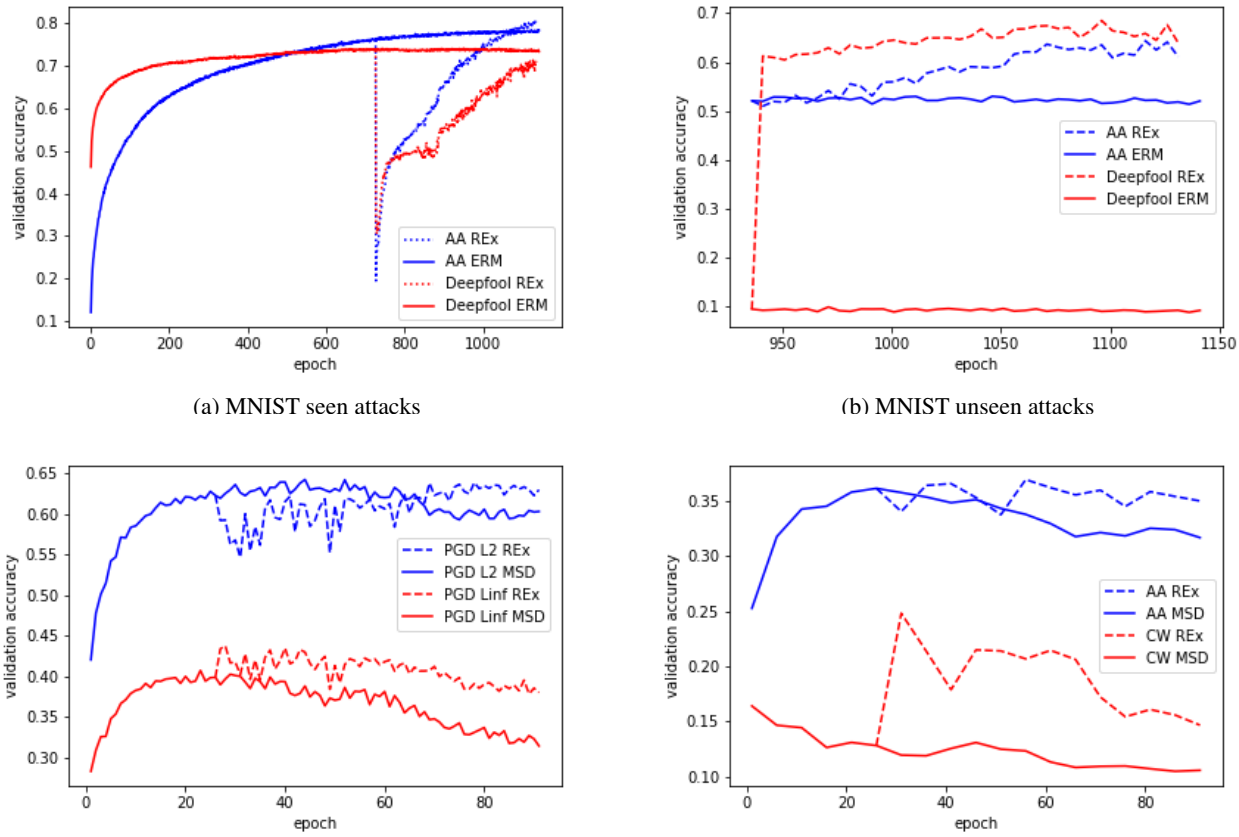
attacks, whether seen or unseen, and in particular on P_∞ and AutoAttack. REx also yields notable improvements against all ensembles, seen or unseen, when used on the Avg baselines. Note however that as in domain generalisation, when used on all baselines except MSD, REx sacrifices performance on the 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 appear to rely on gradient masking compared to the baselines, as the accuracy drops to near 0 values for all models, showing that attacks are successfully computed. A second observation is that the MSD baseline performs surprisingly poorly against AutoAttack and P_∞. While we reused the original code of Maini et al. (2020), there could be multiple reasons for this: firstly, we used momentum SGD when the original paper used Adam. Secondly, we did not use the same architecture as Maini et al. (2020) on MNIST. Thirdly, we did not have a learning rate schedule – albeit it does not seem likely that this would cause such a particular degradation of performance for MSD but not the other baselines. Or, in the case of AutoAttack, the MSD model is just failing to learn how to be robust against the attack. This leads to poor performance against all ensembles of attacks, whether seen or unseen, as those include either L_∞ PGD or AutoAttack adversaries.

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 attacks where REx never yields an improvement are L_1 and L_2 PGD, whether they were seen during training or not. Given the relatively good performance of the baselines and REx on those attacks, this is in line with REx's tendency to sacrifice a few percents of accuracy on the best performing domains to improve significantly the performance on the worst performing domains.

While adversarial training on either L_1 or L_2 PGD 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 as Maini et al. (2020) on CIFAR10). Interestingly, the model adversarially trained on L_∞ PGD performs better than the Avg, Avg_{PGDs} and MSD models on the always unseen set of attacks.



(a) MNIST seen attacks

(b) MNIST unseen attacks

(c) CIFAR10 seen attacks

(d) CIFAR10 unseen attacks

Figure 1: 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, ERM denotes the Avg baseline.

4.3. Early stopping and REx

In Figure 1, 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 most domains, prioritising the worst performing domain. 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 unseen attacks and against the ensembles, as reported in Table 1. See appendix A for more details on how REx is used.

5. Conclusion

While good levels of accuracy have been attained when defending against specific attacks, robustness against multiple attacks has been more difficult to achieve Khoury & Hadfield-Menell (2018); Tramèr & Boneh (2019); Maini et al. (2020). It is important to keep in mind that for example, in a fixed threat model, an attacker may realistically use any allowed attack until the attack succeeds. Therefore, mitigating trade-offs is crucial in order to prevent a significant vulnerability to any particular attack. In our work, by leveraging the connection between the problem of robustness against multiple adversarial attacks and that of generalising to various distributional shifts in the data during training and at test time, we illustrate how a simple approach, REx, can be used to obtain significant improvements in performance against both the seen and unseen attacks in a robust sense (that is, against attackers able to use any of the tested attacks on any sample). A major advantage of REx is that it can be used on top of existing baselines for robustness against multiple attacks, without adding computational complexity per iteration. We vary the dataset, architecture, defense baselines, attack types seen during training, attack types and tuning within a same type of attack at test time, and show that REx leads to consistent improvements in every

case. As future work, we will test our methods on attacks that are not gradient-based, attempt more fine-tuning of the defenses, and test how robust the different baselines and REx are against natural perturbations. Furthermore, we plan to investigate other domain generalisation methods, such as Invariant Risk Minimisation (Arjovsky et al., 2019; Ahuja et al., 2020). We encourage the community to further explore connections between domain generalisation and adversarial robustness.

References

Ahuja, K., Shanmugam, K., Varshney, K., and Dhurandhar, A. Invariant risk minimization games. In *International Conference on Machine Learning*, pp. 145–155. PMLR, 2020. URL https://arxiv.org/abs/2002.04692.

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

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.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. 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.

Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I., and Bethge, M. Accurate, reliable and fast robustness evaluation. *Advances in neural information processing systems*, 32, 2019.

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.

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.

Deng, Y., Zheng, X., Zhang, T., Chen, C., Lou, G., and Kim, M. 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.

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.

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., and Song, D. 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.

Goodfellow, I. A research agenda: Dynamic models to defend against correlated attacks. *arXiv preprint arXiv:1903.06293*, 2019.

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. *arXiv:2007.01434*, 2020.

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.

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.

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.

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.

Li, Z., Shi, C., Xie, Y., Liu, J., Yuan, B., and Chen, Y. 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.

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.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. 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.

Qayyum, A., Qadir, J., Bilal, M., and Al-Fuqaha, A. 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.

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.

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.

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

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. $P^\bullet_\infty$: $\epsilon = 0.4$, $n_{iter} = 40$, $\epsilon_{iter} = 0.033$
7. $DF^\bullet_\infty$: $\epsilon = 0.4$, $n_{iter} = 50$
8. $CW^\bullet_2$: *max_iterations* $= 30$, *learning_rate* $= 0.12$, *binary_search_steps* $= 7$
9. *AutoAttack*: $\epsilon = 0.3$, *norm* $=$ "$Linf$"

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. $P^\bullet_\infty$: $\epsilon = \frac{12}{255}$, $n_{iter} = 70$, $\epsilon_{iter} = \frac{2}{255}$
7. $DF^\bullet_\infty$: $\epsilon = \frac{8}{255}$, $n_{iter} = 50$
8. $CW^\bullet_2$: *max_iterations* $= 30$, *learning_rate* $= 0.012$, *binary_search_steps* $= 7$
9. *AutoAttack*: $\epsilon = \frac{8}{255}$, *norm* $=$ "$Linf''$

A.2. More on how the models are trained

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. 1 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 average accuracy on the seen domains peaks. Note that we do this manually by looking at the seen domains' validation curves. REx is triggered on a baseline before the baseline's early stopping epoch, when progress on the seen domains slows.

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

Using the curves in Fig. 1 as an illustration, on CIFAR10, we chose for example to early stop the MSD baseline at epoch 40. As for the MSD+REx model, REx is activated at epoch 26, and is early stopped at epoch 70. *An important precision about Fig. 1 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. 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 where it is set to 4. These numbers 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. This is done to encourage the optimisation dynamics to neglect neither term of the REx loss.