OUT-OF-DISTRIBUTION GENERALIZATION VIA RISK EXTRAPOLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Generalizing outside of the training distribution is an open challenge for current machine learning systems. A weak form of out-of-distribution (OoD) generalization is the ability to successfully interpolate between multiple observed distributions. One way to achieve this is through robust optimization, which seeks to minimize the worst-case risk over convex combinations of the training distributions. However, a much stronger form of OoD generalization is the ability of models to extrapolate beyond the distributions observed during training. In pursuit of strong OoD generalization, we introduce the principle of **Risk Extrapolation (REx)**. REx can be viewed as encouraging robustness over *affine* combinations of training risks, by encouraging strict equality between training risks. We show conceptually how this principle enables extrapolation, and demonstrate the effectiveness and scalability of instantiations of REx on various OoD generalization tasks.

1 INTRODUCTION

Improving the generalization of deep learning models has become a major research topic, with many different threads of research including Bayesian deep learning (Neal, 1996; Gal, 2016), adversarial (Engstrom et al., 2019; Jacobsen et al., 2018) and non-adversarial Hendrycks & Dietterich (2019); Yin et al. (2019) robustness, causality (Arjovsky et al., 2019), and other works aimed at distinguishing statistical features from semantic features (Gowal et al., 2019; Geirhos et al., 2018). While neural networks often exhibit super-human generalization performance on the training distribution, they can be extremely sensitive to minute changes in distribution Su et al. (2019); Engstrom et al. (2017); Recht et al. (2019). This presents a major bottleneck for their practical application.

In this work, we consider **out-of-distribution (OoD) generalization**, where a model must generalize to new distributions at test time without seeing any training data from them. We assume a fixed underlying task, and access to labeled data from multiple training environments. We also assume that variation in these environments is somewhat representative of the variations we will see at test time. However, we also seek to make good predictions even when these variations are extreme in magnitude.

As a motivating example, consider the colored MNIST (CMNIST) dataset from Arjovsky et al. (2019): there are two training environments in which digits are colored either red or green, with the color being either 80% or 90% correlated with the (binary) label, respectively, and a test environment where color is 10% correlated (i.e. 90% *anti*-correlated) with the label. Generalization to the test set can be viewed as extrapolating over possible values of the correlation (0% to 100%) between color and label. Thus seeing minor variation in this correlation in the training environments is a hint that the correlation could vary even more, and even be reversed, at test time. Like the Invariant Risk Minimization (IRM) approach of Arjovsky et al. (2019), our method of **Risk Extrapolation (REx)** promotes generalization when "spurious" features (such as color, in this example) are predictive during training, but not at test time. For example, in the colored MNIST example, a model that bases its predictions on color will actually do worse than random guessing at test time; the correct way to generalize is to use the "stable" features relating to digit shape.

The classic example of spurious features is the use of background features in object recognition. Most pictures of cows appear in pastures, and so common machine learning approaches struggle to classify pictures of cows on the beach (Beery et al., 2018). In order to build models based on stable features, we can use data collected from different naturally occurring environments (such as geographical

locations), with the hope that the natural variations in distribution across environments will highlight spurious features. How spurious features relate to OoD generalization as a whole is currently unclear, although there is evidence that the lack of robustness of current computer vision models is related to the problem of spurious features (Geirhos et al., 2018; Ilyas et al., 2019).

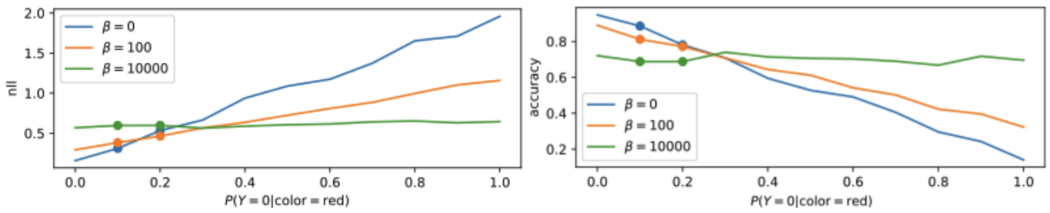

Figure 1: Linear extrapolation of training risks (**left**) and accuracies (**right**) accurately predicts risk and accuracy on colored MNIST test environments with varying $P(Y = 0|\text{color} = \text{red})$. Dots represent training risks, lines represent test risks. Risks are computed after 500 epochs of training. Increasing the V-REx penalty (i.e. β) leads to a flatter "risk plane", which yields more consistent performance across environments. In this case, it also leads the model to focus more on the stable feature of shape, while neglecting the spurious feature of color. This leads to higher risk on environments very similar to the training environments, but better generalization to anti-correlated test environments.

Our approach to finding stable features is inspired by the idea of robust optimization (Ben-Tal et al., 2009), which optimizes the worst-case performance over a set of distributions; for example, this set could be the empirical training distributions collected from different environments. Optimizing the worst-case performance leads the learner to focus entirely on *improving* performance on the *worst*-performing distribution. Our method additionally aims to *decrease* performance on *other* distributions, in order to enforce equal performance across environments. Figure 1 shows how this can lead to better generalization, especially when extrapolating far from the training distributions. We illustrate a trade-off between decreasing risk (i.e. improving performance) on each training environment, and enforcing similarity of risk across environments. When attributes describing the environment change only slightly, lower training risk results in lower test risk (i.e. good generalization). However, as the environment attributes attain values far from those observed in training, similarity between training risks becomes more important for correct generalization. Our method provides a smooth trade-off between these two competing objectives modulated by the hyperparameter β.

To provide a more specific explanation for how performing *worse* on training environments could actually *help* generalization, we return to the colored MNIST example. Since color is more correlated with the label in the second environment than in the first, a predictor which uses color as a feature to predict the labels will do better on the second environment. By encouraging performance to be *equal* across training environments, we can prevent the predictor from using color as a feature, and thus generalize to the test environment where it is anti-correlated with the label.

Our contributions are as follows:

1. We introduce the principle of **Risk Extrapolation (REx)** and explain geometrically and probabilistically how it is able to learn stable features.
2. We derive two practical instantiations of REx. Minimax REx (MM-REx) as a form of robust optimization, and Variance REx (V-REx) minimizing the variance of training risks.
3. We identify crucial considerations for successful training of REx and the related IRM method (Arjovsky et al., 2019).
4. We compare our method to IRM and ERM on a diverse suite of tasks, demonstrating comparable or superior performance in most cases, as well as a limitation of REx in the case of environment-dependent label noise.

2 Background

We consider a multi-environment setting, where our goal is to find parameters θ that perform well on unseen environments, given a set of training environments, $\mathcal{E} = \{e_1, .., e_m\}$. We assume the loss

function, ℓ, and the input and target spaces are fixed across environments. Each environment, e has an associated dataset D_e and data distribution $\mathcal{D}_e$, and a corresponding **risk function**, $\mathbb{R}_e : \mathbb{R}^n \to \mathbb{R}$, which maps θ to the expected loss on that environment's data distribution:

$$\mathbb{R}_e(\theta) \doteq \mathbb{E}_{(x,y)\sim\mathcal{D}_e} \ell_e(f_\theta(x), y) \tag{1}$$

We refer to members of the set $\{\mathbb{R}_e | e \in \mathcal{E}\}$ as the **training risks** or simply **risks**, and we use $\mathbb{R}_i$ to refer to the risk on the i-th environment (and similarly for D_i and $\mathcal{D}_i$).

The **Empirical Risk Minimization (ERM)** approach simply minimizes the average loss across all training examples, regardless of environment:

$$\mathbb{R}_{\text{ERM}}(\theta) \doteq \mathbb{E}_{(x,y)\sim\cup_{e\in\mathcal{E}} D_e}\ \ell(f_\theta(x), y) \tag{2}$$

Empirical risk minimization has strong theoretical foundations in the case of i.i.d. data (Vapnik, 1992), and often works well in practice, outperforming more sophisticated methods (Chen et al., 2019). However it can fail dramatically when test environments differ significantly from training environments (Tzeng et al., 2017).

Robust optimization (RO) (Ben-Tal et al., 2009) is another approach to solving the above problem, designed to hangle distribuional shift. Robust optimization minimizes the worst performance over a set of possible environments $\mathcal{F}$, that is, $\max_{e\in\mathcal{F}} \mathbb{R}_e(\theta)$. Given a single training environment, a set of possible environments is often generated using some set of allowed perturbations, e.g. $P_e(x)$ varying within a KL-divergence ϵ-ball of the empirical distribution (Bagnell, 2005)). When multiple training environments are available, a straightforward approach is to use the empirical distribution of *environments*, setting $\mathcal{F} \doteq \mathcal{E}$:

$$\mathbb{R}_{\text{RO}}(\theta) \doteq \max_{e\in\mathcal{E}} \mathbb{R}_e(\theta) - r_e \tag{3}$$

where r_e is an optional "baseline" risk for each environment, which could represent the difficulty or irreducible risk of that environment.

Invariant Risk Minimization (IRM) (Arjovsky et al., 2019) searches for an invariant representation of inputs from different environments. The IRM principle states: *"An invariant representation $\Phi(X)$ is one such that the optimal linear predictor, w is the same across all environments, e."* Robust optimization may fail to be invariant in this sense, since it may make use of features which are differentially useful across environments (such as color in CMNIST). This leads Arjovsky et al. (2019) to reject this minimax approach in favor of a bilevel optimization problem formulation of the IRM principle:

$$\begin{aligned} &\min_{\Phi, w} \sum_e \mathbb{R}_e(w^\top \Phi(X^e)) \\ &\text{s.t. } w \in \arg\min_{\bar{w}} \mathbb{R}_e(\bar{w}^\top \Phi(X^e)), \quad \forall e \in \mathcal{E} \end{aligned} \tag{4}$$

Arjovsky et al. (2019) also propose **IRMv1** as a practical algorithm for solving the IRM problem:

$$\min_{\Phi} \sum_e \mathbb{R}_e(\Phi(X^e)) + \lambda \|\nabla_w \mathbb{R}_e(w^\top \Phi(X^e)\|_2^2 \tag{5}$$

in which w is a fixed vector of ones, and λ controls the strength of the penalty term on gradients on w. Large gradiens on w indicate that w is not optimal for some environment. We frequently refer to "IRMv1" as simply "IRM". Despite being a form of, REx is more similar to IRM in spirit than Eqn. 3, as we share their motivation of discovering stable features as a means of generalizing OoD. At the same time, although IRM and REx behave quite similarly in most of our experiments, it is easy to show that satisfying the IRM objective does not necessarily lead to the risk across environments being equal. For instance, the IRM solution to the simplest structural equation model in Arjovsky et al. (2019) is an invariant prediction rule for which the risks *do* differ across environments, due to the environment-dependent noise on the targets.

3 METHODS

We now explain and motivate the principle of risk extrapolation, which has two goals:

1. Reducing training risks
2. Increasing similarity of training risks

In general, these goals can be at odds with each other; decreasing the risk in the environment with the lowest risk also decreases the overall similarity of training risks. In this section, we explain how sacrificing performance on *individual* environments in order to perform more similarly *across* environments can help with OoD generalization, and present two methods, **Minimax REx** and **Variance REx**, for achieving, and balancing, goals (1) and (2).

3.1 MINIMAX REX

Here we present a robust optimization method that achieves the aims of risk extrapolation. We begin by observing that we can frame the robust optimization objective from eqn. 3 as a minimax objective over convex combinations of training risks:

$$\mathbb{R}_{\text{RI}}(\theta) \doteq \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq 0}} \sum_e \lambda_e \mathbb{R}_e(\theta) \tag{6}$$

Naturally, we refer to this objective as **Risk Interpolation (RI)**. To arrive at **Minimax REx (MM-REx)**, we relax this objective to allow a more general set of *affine* combinations of training risks:

$$\mathbb{R}_{\text{MM-REx}}(\theta) \doteq \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq -\beta}} \sum_{e=1}^{m} \lambda_e \mathbb{R}_e(\theta) \tag{7}$$

$$= (1 + m\beta) \max_e \mathbb{R}_e(\theta) - \beta \sum_{e=1}^{m} \mathbb{R}_e(\theta)\,, \tag{8}$$

where m is the number of environments, and the hyperparameter β controls how much we seek to extrapolate in the space of risk functions. As $\beta \to \infty$, this criterion enforces strict equality between training risks. If β is negative, it is simply an interpolation coefficient between RI ($\beta = 0$) and ERM ($\beta = -1/m$).

However, for positive values of β, this criterion places negative weights on the risk of all but the worst-case environment. The resulting $\mathbb{R}_{\text{MM-REx}}$ is an *extrapolation* of the training risks, exaggerating variations in the training distributions beyond what has been observed. Larger values of β correspond to extrapolating farther from the convex hull of the training risks, and thus encourage a flatter "risk-plane" (see Figure 1). Note that the constraint that $\sum_i \lambda_i = 1$ means that there is always some pressure to decrease the average risk (but only by aggressively decreasing the highest risk). In practice, we found that this objective would typically enter a regime in which risks are close enough that *which* risk is highest rapidly changes between iterations, and a gradual decrease of average risk can take place.

3.2 REX VIA REGULARIZED OPTIMIZATION

A simpler approach that we find works well in practice is to use the average mean squared error (MSE) between risks, or equivalently, their variance. This yields **Variance REx (V-REx)**, which is the algorithm we focus on in our experiments:

$$\mathbb{R}_{\text{V-REx}} \doteq \sum_{e=1}^{m} \mathbb{R}_e + \beta Var(\mathbb{R}_e) \tag{9}$$

Here $\beta \in [0, \infty)$ controls the balance between reducing average risk and enforcing equality of risks, with $\beta = 0$ recovering ERM, and $\beta \to \infty$ leading V-REx to focus entirely on making the risks equal. We plot the gradient vector field for the total risk as a function of two training risks for both V-REx and MM-REx in Figure A in Appendix A, and note that the gradients change smoothly at the diagonal for $\mathbb{R}_{\text{V-REx}}$, where risks are equal, but not for $\mathbb{R}_{\text{MM-REx}}$.

4 EXPERIMENTS

We evaluate REx and compare with IRM on a range of tasks that require OoD generalization, and investigate the challenges of training these methods. REx performs comparably or better than IRM

on a wide range of tasks, including 1) the PACS domain generalization image dataset, 2) continuous control tasks corrupted with spurious features, 3) predicting financial indicators (see appendix D), and 4) the tasks proposed in Arjovsky et al. (2019). The one exception is for "heteroskedastic" tasks, in which the amount of intrinsic noise in the targets varies across environments. Equalizing risks across these environments does not make sense because the loss in environments with noisier labels will remain higher even for the optimal stable predictor.

4.1 METHODOLOGY FOR OOD

Out-of-distribution generalization (OoD) presents unique challenges for experimental design. In contrast to many commonly-used machine learning benchmarks, in the case of OoD, we cannot assume access to the test *distribution*. This makes it hard to tune hyperparameters in a principled way and care must be taken to not accidentally leak information about the test distribution into this process.

The main challenge is to identify a suitable validation distribution. By definition one should not have access to any distribution that is very close to the test distribution. On the other hand, it is also important to have access to a validation distribution that is sufficiently different from the training distribution, as it is otherwise impossible to properly tune for good OoD generalization.

To resolve the issue in this work, we clearly explain why and when we chose to tune on the test set (e.g. illustrative experiments). In all other cases we strive for good validation practices (e.g. tune on VLCS and apply to PACS).

4.2 CMNIST

Arjovsky et al. (2019) construct a binary classification problem (with 0-4 and 5-9 each collapsed into a single class) based on the MNIST dataset, using color as a spurious feature. Specifically, digits are either colored red or green, and there is a strong correlation between color and label, which is reversed at test time. The goal is to learn the stable "digit shape" feature and ignore the unstable "digit color" feature. The learner has access to three environments:

1. A training environment where green digits have a 80% chance of belonging to class 1 (digits 5-9).
2. A training environment where green digits have a 90% chance of belonging to class 1.
3. A test environment where green digits have a 10% chance of belonging to class 1.

We use the exact same hyperparameters as Arjovsky et al. (2019), only replacing the IRMv1 penalty with MM-REx or V-REx penalty. These methods achieve similar performance, see Table 1.

We emphasize, however, that this task is methodologically unsuited for benchmarking, since, following Arjovsky et al. (2019) we assume access to the test distribution for hyperparameter tuning. Ultimately, these experiments should be interpreted *only* as a demonstration that, unlike ERM or RO, REX and IRM are both capable of OoD generalization in the face of spurious features, when properly tuned.

Method	**train envs acc**	**test env acc**
REx (V-REx) (ours)	71.5 ± 1.0	~~**68.7 ± 0.9**~~
IRM	70.8 ± 0.9	~~66.9 ± 2.5~~
REx (MM-REx) (ours)	72.4 ± 1.8	~~66.1 ± 1.5~~
RI	88.9 ± 0.3	~~22.3 ± 4.6~~
ERM	87.4 ± 0.2	~~17.1 ± 0.6~~
ERM, grayscale (oracle)	73.5 ± 0.2	~~73.0 ± 0.4~~
Theoretical Optimum	75	75
Random Guessing	50	50

Table 1: Accuracy (percent) of different method on the Colored MNIST task. REx and IRM learn to discard spurious color feature and rely on stable shape feature. We use ~~strikethrough~~ to denote results achieved via tuning on the test set.

4.2.1 HOW TO LEARN STABLE FEATURES

In order to begin addressing this issue, we run a series of experiments aimed at understanding the sensitivity of IRM and REx to the choice of hyperparameters. In particular, we note that increasing the relative weight of the penalty term after 100 epochs of training (using a so-called "waterfall" schedule (Desjardins et al., 2015)) is critically important to performance on the colored MNIST task, see Figure 2(b). In light of this finding, we hypothesize that successful learning of stable features using REx or IRM should proceed in two stages. In the first stage, predictive features are learned. In the second stage, stable features are selected and/or predictive features are fine-tuned for stability. This viewpoint suggests that we could use overfitting on the *training* tasks as an indicator for when to apply (or increase) the IRM or REx penalty. This insight could provide a methodologically sound way of tuning the penalty term of REx or IRMv1 in the absence of a representative validation set.

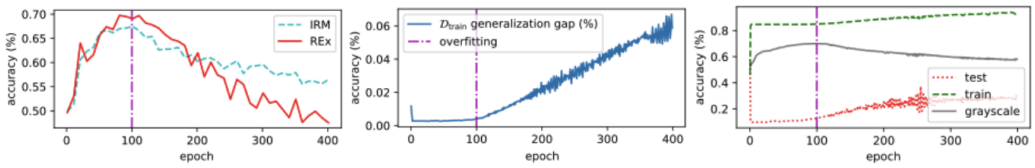

Figure 2: Stability penalties should be applied around when traditional overfitting begins, to ensure that the model has learned predictive features, and that penalties still give meaningful training signals. **Left:** Test accuracy as a function of epoch at which penalty term weight is increased (learning rate is simultaneously decreased proportionally). Choosing this hyperparameter correctly is essential for good performance. **Middle:** Generalization gap on a validation set with 85% correlation between color and label (the same as the average training correlation). The best test accuracy is achieved by increasing the penalty when the generalization gap begins to increase. The increase clearly indicates memorization because color and shape are only 85%/75% correlated with the label, and so cannot be used to make predictions with higher than 85% accuracy. **Right:** Accuracy on training/test sets, as well as an auxilliary grayscale set. Training/test performance reach 85%/15% after a few epochs of training, but grayscale performance improves, showing that meaningful features are still being learned.

In this section, we provide additional support for our hypothesis. In Figure 2, we demonstrate that the optimal point to apply the waterfall is after predictive features have been learned, but before the model starts to memorize training examples. Before predictive features are available, the penalty terms push the model to learn a constant predictor, impeding further learning. And after the model starts to memorize, it become difficult to distinguish spurious and stable features. This second effect is because neural networks often have the capacity to memorize all training examples given sufficient training time, achieving and near-0 loss (Zhang et al., 2016). In the limits of this memorization regime, the differences between losses become small, and gradients of the loss typically do as well, and so the REx and IRMv1 penalties no longer provide a strong or meaningful training signal, See Figure 6 in Appendix C for plot of ERM minimizing REx and IRMv1 penalty terms on Colored MNIST (without including either term in the loss function).

4.3 DOMAIN GENERALIZATION: VLCS AND PACS

In this section we compare REx, IRM and ERM performance on the VLCS (Torralba & Efros, 2011) and PACS (Li et al., 2017) image datasets. Both datasets are commonly-used for multi-source domain generalization. The task is to train on three domains and generalize to a fourth one at test time. For VLCS, the domains are To avoid tuning our hyperparameters on the test distribution, we resort to tuning hyperparameters on VLCS and apply the best setting to PACS without any further tuning.

We use the exact same architecture, training procedure and data augmentation strategy as the state-of-the-art Jigsaw Puzzle approach (Carlucci et al., 2019) (except with IRM or V-REx instead of JigSaw as auxilliary loss) for all three methods. As runs are very noisy, we ran each experiment 10 times, and report average test accuracies extracted at the time of the highest validation accuracy on each run.

On PACS we find that REx outperforms IRM and IRM outperforms ERM on average, while all are worse than the state-of-the-art Jigsaw method.

PACS	Art Painting	Cartoon	Sketch	Photo	Average
REx (ours)	67.04	67.97	59.81	89.74	71.14
IRM	67.05	68.49	57.81	89.39	70.69
ERM	66.22	67.59	57.90	89.69	70.35
Jigsaw (SOTA)	67.43	69.49	62.74	89.64	72.32

Table 2: Accuracy (percent) of different methods on the PACS task. Results are test accuracy at the time of the highest validation accuracy, averaged over 10 runs. REx outperforms ERM and IRM on average, but performs worse than the state-of-the-art.

4.4 Reinforcement Learning with Spurious Correlations

We take tasks from the Deepmind Control Suite Tassa et al. (2018) and modify the original state, $\mathbf{s}$, to include noise and spurious features. Specifically, a scaled copy of 1 or 2 of the dimensions of $\mathbf{s}$ is concatenated with $\mathbf{s}$, and then $\mathcal{N}(0, 0.01)$ noise is added to the corresponding dimensions of the original state vector, to form a new representation of the state, $\bar{\mathbf{s}}$. The magnitude of the scaling factor differs across environments, with $n = 1$ and $n = 2$ for the two training environments. The agent takes $\bar{\mathbf{s}}$ as input and learns a single representation that is shared across training environments. The representation is used by the Soft Actor-Critic Haarnoja et al. (2018) policy learning algorithm, as well as an auxiliary reward predictor, which is trained to predict the next 3 rewards conditioned on the next 3 actions. The representation is trained using gradients from the reward predictor, as well as the critic. The entire model is evaluated on the unseen test environment, which uses a larger scaling factor $n = 3$. Like for CMNIST, since the spurious features are copied from the state before the noise is added, they are more informative for the reward prediction task. In Figure 3, we see that REx outperforms both IRM and ERM in unseen evaluation environments. We use `cartpole_swingup` as a development set for tuning hyperparameters, and perform a single evaluation with 10 random seeds on each of `finger_spin` and `walker_walk`. We tune hyperparameters on `cartpole_swingup` and apply to `finger_spin` and `walker_walk`.

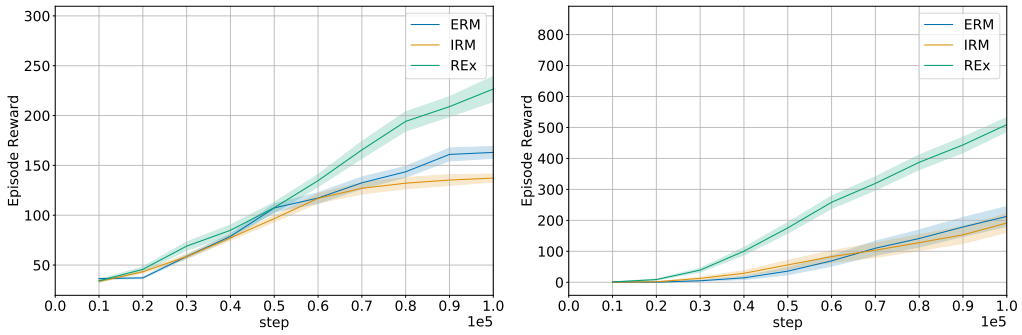

Figure 3: Performance on `walker_walk` (**left**) and `finger_spin` (**right**), with 10 seeds and 1 standard error shaded.

5 Related Work

Out-of-distribution generalization is typically tackled by pre-specifying or learning invariances that aid generalization far outside of the training distribution. There are many approaches to OoD generalization; one is robust learning in the face of worst-case perturbations, another one is causal discovery, which aims to determine an underlying causal graph, and therefore achieve extrapolation. Other approaches are broadly termed domain generalization and often incorporate prior knowledge about the overall problem structure into the training procedure. The assumptions made and approaches proposed can be quite different in these various settings.

One direction in domain generalization is to explicitly project out superficial domain-specific statistics (Wang et al., 2019) to reduce sensitivity to the domain. Another approach increases sensitivity to

shape features of natural images like global spatial coherence of objects (Carlucci et al., 2019) and therefore implicitly reduces reliance on local and superficial cues. Other approaches focus on adversarially learning representations that are invariant with respect to domain-specific features (Tzeng et al., 2017; Long et al., 2018; Li et al., 2018; Albuquerque et al., 2019). Rothenhäusler et al. (2018) note that OoD generalization can be viewed as the solution to a minimax of the worst case risk under different distribution shifts.

Recently, invariance as a proxy for causal inference has been proposed as a way of learning a causal graph. Peters et al. (2016) introduce an algorithm called Invariant Causal Prediction (ICP), to obtain the *causal feature set* (i.e. the minimal set of features that are causal predictors of a target), by exploiting the invariance property that causal models have Pearl (2009); Schölkopf et al. (2012). Arjovsky et al. (2019) propose an extension of this work called invariant risk minimization (IRM), in which empirical risk minimization is augmented to learn a data representation that is free of spurious correlations.

Others in causal inference propose disentanglement as a way to uncover and separate causal variables from spurious correlations. Heinze-Deml & Meinshausen (2017) assume "grouping" knowledge, that one has access to the same object under varying conditions, in order to disentangle what features are invariant conditional on a target variable, and what are simply "style". Similarly, Gowal et al. (2019) attempt to find a representation in which "style" (i.e. spurious) and "semantic" (i.e. stable) features are disentangled. This allows them to manipulate the style of training images as a form of adversarial training (Madry et al., 2017) with semantic perturbations.

6 Discussion

While the high level goals of IRM and REx are similar, IRM has a much more developed mathematical foundation. On the other hand, REx has simple geometric and probabilistic interpretations, which may be more intuitive.

Empirically, REx and IRM perform on par in many settings. However, on the PACS domain generalization benchmark REx slightly outperforms IRM and in our reinforcement learning experiments we observe a clear advantage of REx over IRM. Despite the encouraging empirical performance, REx has a shortcoming when compared to IRM. It may not succeed across environments with different levels of intrinsic label noise, whereas IRM can deal with such fluctuations.

Thus, a key unresolved challenge for REx is how to fairly compare risk functions across environments that contain different levels of innate noise, since it may not make sense to enforce exact equality of risks in that case. The use of "baselines" designed to account for these differences exists in the literature (Meinshausen et al., 2015), but the baseline proposed by Meinshausen et al. (2015), $Var(Y)$, only provides an upper bound on the true noise-level.

7 Conclusion

We have introduced risk extrapolation as a new method to generalize outside of the training distribution. Our method is a simple generalization of the idea underlying robust optimization, inspired by Invariant Risk Minimization, and their goal of learning stable features. We have shown that maximizing the overall risk over a specified set of affine combinations of the training risks means that all training risks should be made as similar as possible. This can be implemented efficiently with a variance penalty on the environment risks. We show that our method performs competitively on a range of out-of-distribution and domain generalization tasks.

References

Isabela Albuquerque, João Monteiro, Tiago H Falk, and Ioannis Mitliagkas. Adversarial target-invariant representation learning for domain generalization. *arXiv preprint arXiv:1911.00804*, 2019.

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

J. Andrew Bagnell. Robust supervised learning. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, pp. 714–719. AAAI Press, 2005. ISBN 157735236x.

Sara Beery, Grant Van Horn, and Pietro Perona. Recognition in terra incognita. *Lecture Notes in Computer Science*, pp. 472–489, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01270-0_28. URL http://dx.doi.org/10.1007/978-3-030-01270-0_28.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton University Press, 2009.

Fabio M Carlucci, Antonio D'Innocente, Silvia Bucci, Barbara Caputo, and Tatiana Tommasi. Domain generalization by solving jigsaw puzzles. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2229–2238, 2019.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*, 2019.

Guillaume Desjardins, Karen Simonyan, Razvan Pascanu, et al. Natural neural networks. In *Advances in Neural Information Processing Systems*, pp. 2071–2079, 2015.

Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. *arXiv preprint arXiv:1712.02779*, 2017.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. Learning perceptually-aligned representations via adversarial robustness. *arXiv preprint arXiv:1906.00945*, 2019.

Yarin Gal. Uncertainty in deep learning. 2016.

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

Sven Gowal, Chongli Qin, Po-Sen Huang, Taylan Cemgil, Krishnamurthy Dvijotham, Timothy Mann, and Pushmeet Kohli. Achieving robustness in the wild via adversarial mixing with disentangled representations. *arXiv preprint arXiv:1912.03192*, 2019.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

Christina Heinze-Deml and Nicolai Meinshausen. Conditional variance penalties and domain shift robustness, 2017.

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

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. In *Advances in Neural Information Processing Systems*, pp. 125–136, 2019.

Jörn-Henrik Jacobsen, Jens Behrmann, Richard Zemel, and Matthias Bethge. Excessive invariance causes adversarial vulnerability. *arXiv preprint arXiv:1811.00401*, 2018.

Da Li, Yongxin Yang, Yi-Zhe Song, and Timothy M Hospedales. Deeper, broader and artier domain generalization. In *Proceedings of the IEEE international conference on computer vision*, pp. 5542–5550, 2017.

Ya Li, Xinmei Tian, Mingming Gong, Yajing Liu, Tongliang Liu, Kun Zhang, and Dacheng Tao. Deep domain generalization via conditional invariant adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 624–639, 2018.

Mingsheng Long, Zhangjie Cao, Jianmin Wang, and Michael I Jordan. Conditional adversarial domain adaptation. In *Advances in Neural Information Processing Systems*, pp. 1640–1650, 2018.

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

Nicolai Meinshausen, Peter Bühlmann, et al. Maximin effects in inhomogeneous large-scale data. *The Annals of Statistics*, 43(4):1801–1830, 2015.

Radford M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.

Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, New York, NY, USA, 2nd edition, 2009. ISBN 052189560X, 9780521895606.

Jonas Peters, Peter Bühlmann, and Nicolai Meinshausen. Causal inference by using invariant prediction: identification and confidence intervals. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 78(5):947–1012, 2016.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? *arXiv preprint arXiv:1902.10811*, 2019.

Dominik Rothenhäusler, Nicolai Meinshausen, Peter Bühlmann, and Jonas Peters. Anchor regression: heterogeneous data meets causality, 2018.

Bernhard Schölkopf, Dominik Janzing, Jonas Peters, Eleni Sgouritsa, Kun Zhang, and Joris Mooij. On causal and anticausal learning. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 459–466, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841, 2019.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy Lillicrap, and Martin Riedmiller. DeepMind control suite. Technical report, DeepMind, January 2018.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *CVPR 2011*, pp. 1521–1528. IEEE, 2011.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. Adversarial discriminative domain adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7167–7176, 2017.

V. Vapnik. Principles of risk minimization for learning theory. In J. E. Moody, S. J. Hanson, and R. P. Lippmann (eds.), *Advances in Neural Information Processing Systems 4*, pp. 831–838. Morgan-Kaufmann, 1992.

Haohan Wang, Zexue He, Zachary C Lipton, and Eric P Xing. Learning robust representations by projecting superficial statistics out. *arXiv preprint arXiv:1903.06256*, 2019.

Dong Yin, Raphael Gontijo Lopes, Jon Shlens, Ekin Dogus Cubuk, and Justin Gilmer. A fourier perspective on model robustness in computer vision. In *Advances in Neural Information Processing Systems*, pp. 13255–13265, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

A APPENDIX: GRADIENT VECTOR FIELDS OF REX

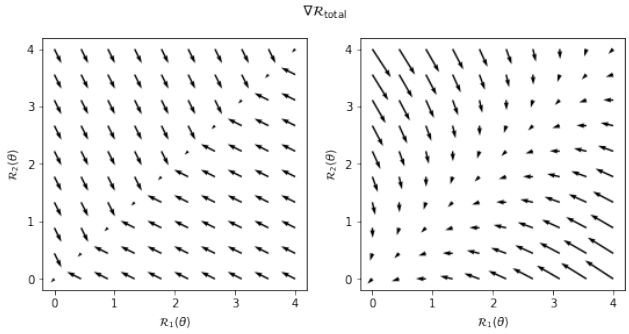

Figure 4: Vector fields of the gradient evaluated at different values of training risks $\mathbb{R}_1(\theta)$, $\mathbb{R}_2(\theta)$. We compare the gradients for $\mathbb{R}_{\text{MM-REx}}$ (**left**) and $\mathbb{R}_{\text{V-REx}}$ (**right**). Note that for $\mathbb{R}_{\text{V-REx}}$, the gradient vectors curve smoothly towards the direction of the origin, as they approach the diagonal (where training risks are equal); this leads to a smoother optimization landscape.

B APPENDIX: A PROBABILISTIC VIEW ON RISK EXTRAPOLATION

Since the operation of integration is a linear functional, and since we assume a fixed loss function, we have $\mathbb{R}(\theta) = \int_{x,y} P(x,y)\ell(f_\theta(x), y)$ is linear in $P(x,y)$. This means we can understand risk extrapolation "term-wise" by looking at its effects at every point $\{x, y\}$. In particular, linear combinations of risk functions can be understood in terms of the corresponding linear combinations of probability density functions (PDFs). Figure B shows a graphical example of how REx allows us to effectively synthesize environments with distributions very different from those of the training environments, in a way that risk interpolation does not, by using negative weights on the training risks.

While convex combinations of PDFs yield mixture distributions, a non-convex combination will not generally be a distribution, itself. *Affine* combinations *do* maintain the constraint that their integral is equal to 1, but may also assign "negative probabilities" to some points. We reinterpret these negative probabilities as a positive probability of the same example occurring, but with the *loss function* negated. The equivalence is exact, and results from simultaneously negating $P(x,y)$ and $\ell(f_\theta(x), y)$ for the corresponding terms in the integral.

Since typical loss functions, such as mean-squared error and negative log-likelihood, can take on unboundedly high values, negating the loss function could hypothetically lead to unboundedly low loss. However, this is not a concern for the REx methods we propose, since this would also correspond to a larger gap in the training risks, which would be penalized. In particular, $\mathbb{R}_{\text{MM-REx}} \geq \mathbb{R}_{\text{ERM}}$, since the $\max$ in $\mathbb{R}_{\text{MM-REx}}$ could always set the $\lambda_i = 1/m$ to produce the ERM risk. Meanwhile, $\mathbb{R}_{\text{V-REx}}$ penalizes the difference of an individual training risk from the mean risk *quadratically*, outweighing the linear reduction in mean risk that assigning a negative weight to some examples allows.

Overall, the net effect of such negative probabilities is to prevent the model from making predictions that are overconfident (in classification) or extreme (in regression) on the basis of features that vary across environments, since doing so would lead to differences in the training risks that would be heavily penalized.

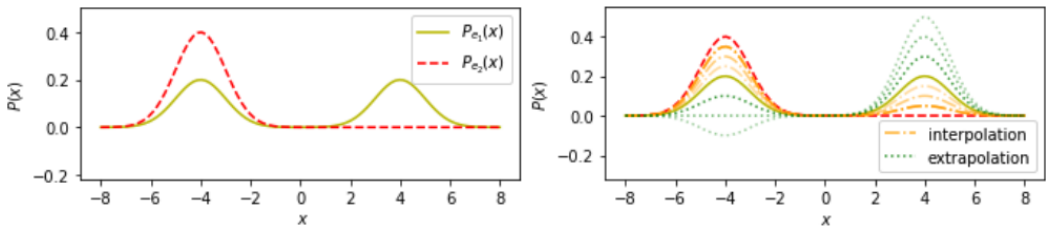

Figure 5: The probabilistic view of risk extrapolation, illustrated. **Left:** Input distributions $P(x)$ for environments e_1 and e_2. **Right Top:** Interpolating between risk functions $\mathbb{R}_1$ and $\mathbb{R}_2$ corresponds to point-wise interpolations of $P_{e_1}(x)$ and $P_{e_2}(x)$. **Right Bottom:** Extrapolation can yield a distribution with zero, or even *negative* probability for examples from the left mode. These negative probabilities correspond to changes in $P(y|x)$, e.g. positive probabilities for examples with flipped labels.

C Appendix: Memorization (via ERM) is a (bad) additional way to minimize IRMv1 and REx penalties

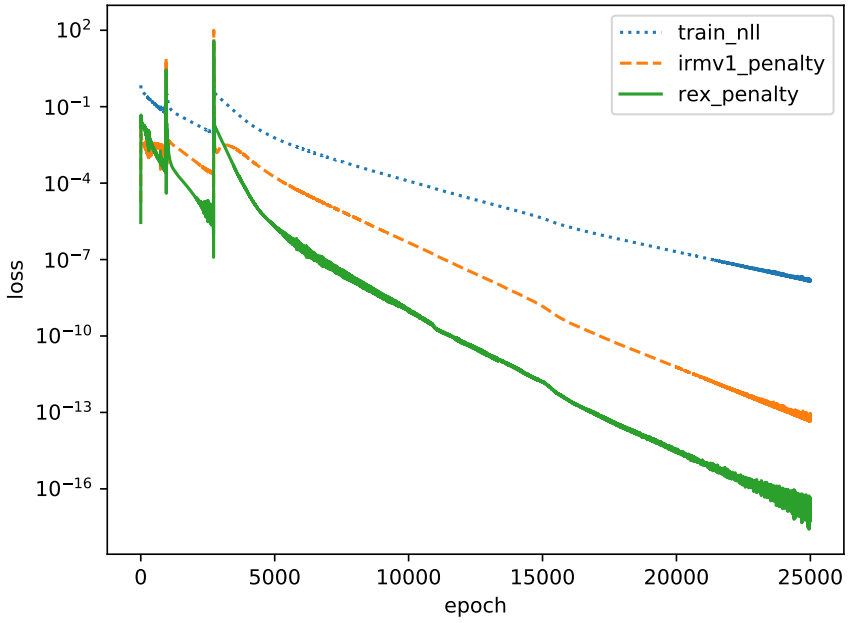

Figure 6: Given sufficient training time, empirical risk minimization (ERM) minimizes both REx and IRMv1 penalty terms on Colored MNIST (*without* including either term in the loss function). This is because the model (a deep network) has sufficient capacity to fit the training sets almost perfectly. This prevents these penalties from having the intended effect, once the model has started to overfit. The y-axis is in log-scale.

D Appendix: Financial indicators

To compare the types of solutions found by IRM and REx, we apply our method to a prediction task in the finance domain, where countless confounding and unknown factors make prediction based on a small number of features hard-to-impossible. The dataset is split into five years, 2014–18, containing 37 publicly reported financial indicators of several thousand publicly listed companies each. The task is to predict if a company's value will increase or decrease in the following year. We

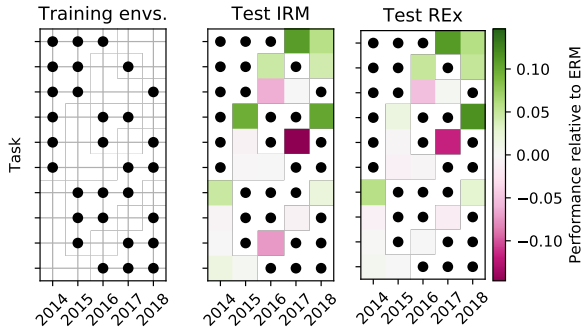

Figure 7: Financial indicators tasks. The left panel indicates the set of training environments; the middle and right panels show the test accuracy on the respective environments relative to ERM (a black dot corresponds to a training environment; a colored patch indicates the test accuracy on the respective environment.)

consider each year a different environment, and create 20 different tasks by selecting all possible combinations of environments where three environments represent the training sets, one environment the validation set, and another one the test set. We train an MLP using the validation set to determine an early stopping point, with $\beta = 10^4$. The per-task results summarized in fig. 7 indicate substantial differences between ERM and IRM, and ERM and REx. The predictions produced by IRM and REx, however, only differ insignificantly, highlighting the similarity of IRM and REx.