Appendices

A. Appendix Overview

Our code is available online at: https://anonymous.4open.science/r/12747e81-8505-43cb-b54e-e75e2344a397/. The sections of our appendix are as follows:

B. Definition and discussion of extrapolation in machine learning

We define interpolation and extrapolation as follows: **interpolation** refers to making decisions or predictions about points/domains *within* the convex hull of the training examples/domains and **extrapolation** refers to making decisions or predictions about points/domains *outside* their convex hull.[14] This generalizes the familiar sense of these terms for one-dimensional functions. An interesting consequence of this definition is: for data of high intrinsic dimension, generalization *requires* extrapolation (Hastie et al., 2009), even in the i.i.d. setting. This is because the volume of high-dimensional manifolds concentrates near their boundary; see Figure 7.

Extrapolation in the space of risk functions. The same geometric considerations apply to extrapolating to new domains. Domains can be highly diverse, varying according to high dimensional attributes, and thus requiring extrapolation to generalize across. Thus Risk Extrapolation might often do a better job of including possible test domains in its perturbation set than Risk Interpolation does.

[14]Surprisingly, we were not able to find any existing definition of these terms in the machine learning literature, although this definition is proposed by King & Zeng (2006). They have also been used in this sense (Hastie et al., 2009; Haffner, 2001), but also to refer to strong generalization capabilities more generally (Sahoo et al., 2018; Xu et al., 2021). Meanwhile, Mouli & Ribeiro (2021) argue that extrapolation should be thought of in terms of generalizing to counterfactual domains; our view shows how this can sometimes be viewed as extrapolating outside the convex hull in "domain space".

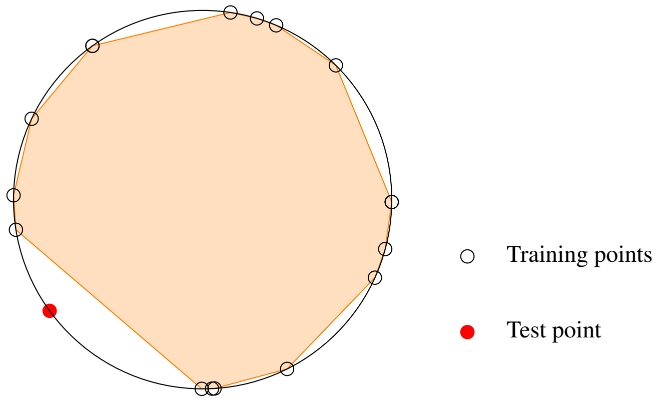

Figure 7. Illustration of the importance of extrapolation for generalizing in high dimensional space. In high dimensional spaces, mass concentrates near the boundary of objects. For instance, the uniform distribution over a ball in $N + 1$-dimensional space can be approximated by the uniform distribution over the N-dimensional hypersphere. We illustrate this in 2 dimensions, using the 1-sphere (i.e. the unit circle). Dots represent a finite training sample, and the shaded region represents the convex hull of all but one member of the sample. Even in 2 dimensions, we can see why any point from a finite sample from such a distribution remains outside the convex hull of the other samples, with probability 1. The only exception would be if two points in the sample coincide *exactly*.

C. Illustrative examples of how REx works in toy settings

Here, we work through two examples to illustrate:

1. How to understand extrapolation in the space of probability density/mass functions (PDF/PMFs)
2. How REx encourages robustness to covariate shift via distributing capacity more evenly across possible input distributions.

C.1. 6D example of REx

Here we provide a simple example illustrating how to understand extrapolations of probability distributions. Suppose $X \in \{0, 1, 2\}$ and $Y \in \{0, 1\}$, so there are a total of 6 possible types of examples, and we can represent their distributions in a particular domain as a point in 6D space: $(P(0, 0), P(0, 1), P(1, 0), P(1, 1), P(2, 0), P(2, 1))$. Now, consider three domains e_1, e_2, e_3 given by

1. (a, b, c, d, e, f)
2. $(a, b, c, d, e - k, f + k)$
3. $(2a, 2b, c(1 - \frac{a+b}{c+d}), d(1 - \frac{a+b}{c+d}), e, f)$

The difference between e_1 and e_2 corresponds to a shift in $P(Y|X = 2)$, and suggests that Y cannot be reliably predicted across different domains when $X = 2$. Meanwhile, the difference between e_1 and e_3 tells us that the relative probability of $X = 0$ vs. $X = 1$ can change, and so we might want our model to be robust to these sorts of covariate shifts. Extrapolating risks across these 3 domains effectively tells the model: "don't bother trying to predict Y when $X = 2$ (i.e. aim for $\hat{P}(Y = 1|X = 2) = .5$), and split your capacity equally across the $X = 0$ and $X = 1$ cases". By way of comparison, IRM would also aim for $\hat{P}(Y = 1|X = 2) = .5$, whereas ERM would aim for $\hat{P}(Y = 1|X = 2) = \frac{3f+k}{3e+3f}$ (assuming $|D_1| = |D_2| = |D_3|$). And unlike REx, both ERM and IRM would split capacity between $X = 0/1/2$ cases according to their empirical frequencies.

C.2. Covariate shift example

We now give an example to show how REx provides robustness to covariate shift. Covariate shift is an issue when a model has limited capacity or limited data.

Viewing REx as robust learning over the affine span of the training distributions reveals its potential to improve robustness to distribution shifts. Consider a situation in which a model encounters two types of inputs: COSTLY inputs with probability q and CHEAP inputs with probability $1-q$. The model tries to predict the input – it outputs COSTLY with probability p and CHEAP with probability $1-p$. If the model predicts right its risk is 0, but if it predicts COSTLY instead of CHEAP it gets a risk $u=2$, and if it predicts CHEAP instead of COSTLY it gets a risk $v=4$. The risk has expectation $\mathcal{R}_q(p)=(1-p)(1-q)u+pqv$. We have access to two domains with different input probabilities $q_1<q_2$. This is an example of pure covariate shift.

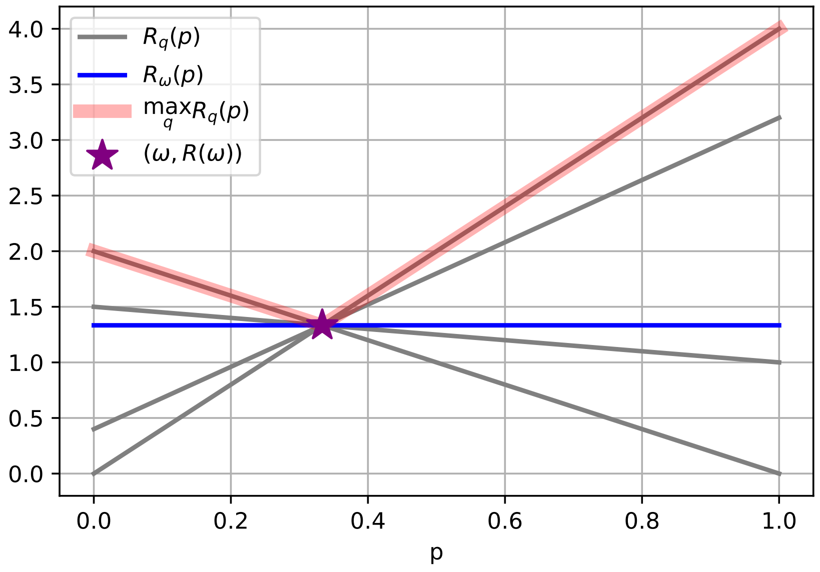

Figure 8. Each grey line is a risk $\mathcal{R}_q(p)$ as functions of p for a specific value of q. The blue line is when $q=\omega$. We highlight in red the curve $\max_q \mathcal{R}_q(p)$ whose minimum is the saddle point marked by a purple star in $p=\omega$.

We want to guarantee the minimal risk over the set of all possible domains:

$$\min_{p\in[0,1]} \max_{q\in[0,1]} \mathcal{R}_q(p)=(1-p)(1-q)u+pqv$$

as illustrated in Figure 8. The saddle point solution of this problem is $p=\omega={}^{u}/_{u+v}$ and $\mathcal{R}_q(p)={}^{uv}/_{u+v}, \forall q$. From the figure we see that $\mathcal{R}_{q_1}(p)=\mathcal{R}_{q_2}(p)$ can only happen for $p=\omega$, so the risk extrapolation principle will return the minimax optimal solution.

If we use ERM to minimize the risk, we will pool together the domains into a new domain with COSTLY input probability $\bar{q}=(q_1+q_2)/2$. ERM will return $p=0$ if $\bar{q}>\omega$ and $p=1$ otherwise. Risk interpolation (RI) $\min_p \max_{q\in\{q_1,q_2\}} R_q(p)$ will predict $p=0$ if $q_1,q_2>\omega$, $p=1$ if $q_1,q_2<\omega$ and $p=\omega$ if $q_1<\omega<q_2$. We see that only REx finds the minimax optimum for arbitrary values of q_1 and q_2.

D. A summary of different types of causal models

Here, we briefly summarize the differences between 3 different types of causal models, see Table 4. Our definitions and notation follow *Elements of Causal Inference: Foundations and Learning Algorithms* (Peters et al., 2017).

A **Causal Graph** is a directed acyclic graph (DAG) over a set of nodes corresponding to random variables $\mathbf{Z}$, where edges point from causes (including noise variables) to effects. A **Structural Causal Model (SCM)**, $\mathfrak{C}$, additionally specifies a *deterministic* mapping f_Z for every node Z, which computes the value of that node given the values of its parents, which include a special noise variable N_Z, which is sampled independently from all other nodes. This f_Z is called the **mechanism**, **structural equation**, or **structural assignment** for Z. Given an SCM, $\mathfrak{C}$, the **entailed distribution** of $\mathfrak{C}$, $P^{\mathfrak{C}}(\mathbf{Z})$ is defined via ancestral sampling. Thus for any $Z \in \mathbf{Z}$, we have that the marginal distribution $P^{\mathfrak{C}}(Z|\mathbf{Z} \setminus Z) = P^{\mathfrak{C}}(Z|Pa(Z))$. A **Causal Graphical Model (CGM)** can be thought of as specifying these marginal distributions *without* explicitly representing noise variables N_Z. We can draw rough analogies with (non-causal) statistical models. Roughly speaking, Causal Graphs are analogous to Graphical Models, whereas SCMs and CGMs are analogous to joint distributions.

Model	Independences	Distributions	Interventions	Counterfactuals
Graphical Model	✓	✗	✗	✗
Joint Distribution	✓	✓	✗	✗
Causal Graph	✓	✗	✓	✗
Causal Graphical Model	✓	✓	✓	✗
Structural Causal Model	✓	✓	✓	✓

Table 4. A comparison of causal and non-causal models.

E. Theory

E.1. Proofs of theorems 1 and 2

The REx principle (Section 3) has two goals:

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

In practice, it may be advantageous to trade-off these two objectives, using a hyperparameter (e.g. β for V-REx or $\lambda_{\min}$ for MM-REx). However, in this section, we assume the 2nd criteria takes priority; i.e. we define "satisfying" the REx principle as selecting a minimal risk predictor among those that achieve *exact* equality of risks across all the domains in a set $\mathcal{E}$.

Recall our assumptions from Section 3.2 of the main text:

1. The causes of Y are observed, i.e. $Pa(Y) \subseteq X$.
2. Domains correspond to interventions on X.
3. Homoskedasticity (a slight generalization of the additive noise setting assumed by Peters et al. (2016)). We say an SCM $\mathfrak{C}$ is **homoskedastic** (with respect to a loss function ℓ), if the Bayes error rate of $\ell(f_Y(x), f_Y(x))$ is the same for all $x \in \mathcal{X}$.

And see Section 2.3 for relevant definitions and notation.

We begin with a theorem based on the setting explored by Peters et al. (2016). Here, $\varepsilon_i \doteq N_i$ are assumed to be normally distributed.

Theorem 1. *Given a Linear SEM, $X_i \leftarrow \sum_{j \neq i} \beta_{(i,j)} X_j + \varepsilon_i$, with $Y \doteq X_0$, and a predictor $f_\beta(X) \doteq \sum_{j:j>0} \beta_j X_j + \varepsilon_j$ that satisfies REx (with mean-squared error) over a perturbation set of domains that contains 3 distinct do() interventions for each $X_i : i > 0$. Then $\beta_j = \beta_{0,j}, \forall j$.*

Proof. We adapt the proof of Theorem 4i from Peters et al. (2016) to show that REx will learn the correct model under similar assumptions. Let $Y \leftarrow \gamma X + \varepsilon$ be the mechanism for Y, assumed to be fixed across all domains, and let $\hat{Y} = \beta X$ be our predictor. Then the residual is $R(\beta) = (\gamma - \beta)X + \varepsilon$. Define $\alpha_i \doteq \gamma_i - \beta_i$, and consider an intervention $do(X_j = x)$ on the youngest node X_j with $\alpha_j \neq 0$. Then as in eqn 36/37 of Peters et al. (2016), we compare the residuals R of this intervention and of the observational distribution:

$$R^{\text{obs}}(\beta) = \alpha_j X_j + \sum_{i \neq j} \alpha_i X_i + \varepsilon \qquad\qquad R^{do(X_j=x)}(\beta) = \alpha_j x + \sum_{i \neq j} \alpha_i X_i + \varepsilon \tag{9}$$

We now compute the MSE risk for both domains, set them equal, and simplify to find a quadratic formula for x:

$$\mathbb{E}\left[(\alpha_j X_j + \sum_{i \neq j} \alpha_i X_i + \varepsilon)^2\right] = \mathbb{E}\left[(\alpha_j x + \sum_{i \neq j} \alpha_i X_i + \varepsilon)^2\right] \tag{10}$$

$$0 = \alpha_j^2 x^2 + 2\alpha_j \mathbb{E}[\sum_{i \neq j} \alpha_i X_i + \varepsilon]x - \mathbb{E}\left[(\alpha_j X_j)^2 - 2\alpha_j X_j (\sum_{i \neq j} \alpha_i X_i + \varepsilon)\right] \tag{11}$$

Since there are at most two values of x that satisfy this equation, any other value leads to a violation of REx, so that α_j needs to be zero – contradiction. In particular having domains with 3 different *do*-interventions on every X_i guarantees that the risks are not equal across all domains. □

Given the assumption that a predictor satisfies REx over *all* interventions that do not change the mechanism of Y, we can prove a much more general result. We now consider an arbitrary SCM, $\mathfrak{C}$, generating Y and X, and let $\mathcal{E}^I$ be the set of domains corresponding to arbitrary interventions on X, similarly to Peters et al. (2016).

We emphasize that the predictor is not restricted to any particular class of models, and is a generic function $f : \mathcal{X} \to \mathcal{P}(Y)$, where $\mathcal{P}(Y)$ is the set of distributions over Y. Hence, we drop θ from the below discussion and simply use f to represent the predictor, and $\mathcal{R}(f)$ its risk.

Theorem 2. *Suppose ℓ is a (strictly) proper scoring rule. Then a predictor that satisfies REx for a over $\mathcal{E}^I$ uses $f_Y(x)$ as its predictive distribution on input x for all $x \in \mathcal{X}$.*

Proof. Let $\mathcal{R}^e(f, x)$ be the loss of predictor f on point x in domain e, and $\mathcal{R}^e(f) = \int_{P^e(x)} \mathcal{R}^e(f, x)$ be the risk of f in e. Define $\iota(x)$ as the domain given by the intervention $do(X = x)$, and note that $\mathcal{R}^{\iota(x)}(f) = \mathcal{R}^{\iota(x)}(f, x)$. We additionally define $X_1 \doteq Par(Y)$.

The causal mechanism, f_Y, satisfies the REx principle over $\mathcal{E}^I$. For every $x \in \mathcal{X}$, $f_Y(x) = P(Y|do(X = x)) = P(Y|do(X_1 = x_1)) = P(Y|X_1 = x_1)$ is invariant (meaning 'independent of domain') by definition; $P(Y|do(X = x)) = P(Y|do(X_1 = x_1)) = P(Y|X_1 = x_1)$ follows from the semantics of SEM/SCMs, and the fact that we don't allow f_Y to change across domains. Specifically Y is always generated by the same ancestral sampling process that only depends on X_1 and N^Y. Thus the risk of the predictor $f_Y(x)$ at point x, $\mathcal{R}^e(f_Y, x) = \ell(f_Y(x), f_Y(x))$ is also invariant, soit $\mathcal{R}(f_Y, x)$. Thus $\mathcal{R}^e(f_Y) = \int_{P^e(x)} \mathcal{R}^e(f_Y, x) = \int_{P^e(x)} \mathcal{R}(f_Y, x)$ is invariant whenever $\mathcal{R}(f_Y, x)$ does not depend on x, and the homoskedasticity assumption ensures that this is the case. This establishes that setting $f = f_Y$ will produce equal risk across domains.

No other predictor satisfies the REx principle over $\mathcal{E}^I$. We show that any other g achieves higher risk than f_Y for at least one domain. This demonstrates both that f_Y achieves minimal risk (thus satisfying REx), and that it is the unique predictor which does so (and thus no other predictors satisfy REx). We suppose such a g exists and construct an domain where it achieves higher risk than f_Y. Specifically, if $g \neq f_Y$ then let $x \in \mathcal{X}$ be a point such that $g(x) \neq f_Y(x)$. And since ℓ is a strictly proper scoring rule, this implies that $\ell(g(x), f_Y(x)) > \ell(f_Y(x), f_Y(x))$. But $\ell(g(x), f_Y(x))$ is exactly the risk of g on the domain $\iota(do(X = x))$, and thus g achieves higher risk than f_Y in $\iota(do(X = x))$, a contradiction. □

E.2. REx as DRO

We note that MM-REx is also performing robust optimization over a convex hull, see Figure 1. The corners of this convex hull correspond to "extrapolated domains" with coefficients $(\lambda_{\min}, \lambda_{\min}, ..., (1 - (m-1)\lambda_{\min}))$ (up to some permutation). However, these domains do not necessarily correspond to valid probability distributions; in general, they are quasidistributions, which can assign negative probabilities to some examples. This means that, even if the original

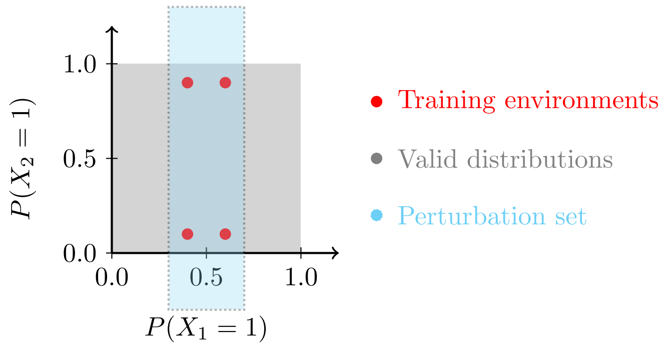

Figure 9. The perturbation set for MM-REx can include "distributions" which assign invalid (e.g. negative) probabilities to some data-points. The range of valid distributions $P(X)$ is shown in grey, and $P(X)$ for 4 different training domains are shown as red points. The interior of the dashed line shows the perturbation set for $\lambda_{\min} = -1/2$.

risk functions were convex, the extrapolated risks need not be. However, in the case where they *are* convex, then existing theorems, such as the convergence rate result of (Sagawa et al., 2019). This raises several important questions:

1. When is the affine combination of risks convex?
2. What are the effects of negative probabilities on the optimization problem REx faces, and the solutions ultimately found?

Negative probabilities: Figure 9 illustrates this for a case where $\mathcal{X} = \mathbb{Z}_2^2$, i.e. x is a binary vector of length 2. Suppose x_1, x_2 are independent in our training domains, and represent the distribution for a particular domain by the point $(P(X_1 = 1), P(X_2 = 1))$. And suppose our 4 training distributions have $(P(X_1 = 1), P(X_2 = 1))$ equal to $\{(.4, .1), (.4, .9), (.6, .1), (.6, .9)\}$, with $P(Y|X)$ fixed.

F. The relationship between MM-REx vs. V-REx, and the role each plays in our work

The MM-REx and V-REx methods play different roles in our work:

- We use MM-REx to illustrate that REx can be instantiated as a variant of robust optimization, specifically a generalization of the common Risk Interpolation approach. We also find MM-REx provides a useful geometric intuition, since we can visualize its perturbation set as an expansion of the convex hull of the training risks or distributions.
- We expect V-REx to be the more practical algorithm. It is simple to implement. And it performed better in our CMNIST experiments; we believe this may be due to V-REx providing a smoother gradient vector field, and thus more stable optimization, see Figure 3.1.

Note that despite our presentation moving from MM-REx to V-REx, we do not view V-REx as an approximation of MM-REx, and we do not suggest or have any reason to believe MM-REx is a better objective in principle. Either method recovers the REx principle as a limiting case, as we prove in Section F.1. We also provide a sequence of mathematical derivations that sheds light on the relationship between MM-REx and V-REx in Section F.2. We can view these as a progression of steps for moving from the robust optimization formulation of MM-REx to the penalty:

1. **From minimax to closed form:** We show how to arrive at the closed-form version of MM-REx provided in Eqn. 7.
2. **Closed form as mean absolute error:** The closed form of MM-REx is equivalent to a mean absolute error (MAE) penalty term when there are only two training domains.
3. **V-REx as mean squared error:** V-REx is exactly equivalent to a mean squared error penalty term (always). Thus in the case of only two training domains, the difference between MM-REx and V-REx is just a different choice of norm.

F.1. V-REx and MM-REx enforce the REx principle in the limit

We prove that both MM-REx and V-REx recover the constraint of perfect equality between risks in the limit of $\lambda_{\min} \to -\infty$ or $\beta \to \infty$, respectively. For both proofs, we assume all training risks are finite.

Proposition 1. *The MM-REx risk of predictor f_θ, $\mathcal{R}_{\mathrm{MM-REx}}(\theta) \to \infty$ as $\lambda_{\min} \to -\infty$ unless $\mathcal{R}^d = \mathcal{R}^e$ for all training domains d, e.*

Proof. Suppose the risk is not equal across domains, and let the largest difference between any two training risks be $\epsilon > 0$. Then $\mathcal{R}_{\mathrm{MM-REx}}(\theta) = (1 - m\lambda_{\min}) \max_e \mathcal{R}_e(\theta) + \lambda_{\min} \sum_{i=1}^{m} \mathcal{R}_i(\theta) = \max_e \mathcal{R}_e(\theta) - m\lambda_{\min} \max_e \mathcal{R}_e(\theta) + \lambda_{\min} \sum_{i=1}^{m} \mathcal{R}_i(\theta) \geq \max_e \mathcal{R}_e(\theta) - \lambda_{\min}\epsilon$, with the inequality resulting from matching up the m copies of $\lambda_{\min} \max_e \mathcal{R}_e$ with the terms in the sum and noticing that each pair has a non-negative value (since $\mathcal{R}_i - \max_e \mathcal{R}_e$ is non-positive and $\lambda_{\min}$ is negative), and at least one pair has the value $-\lambda_{\min}\epsilon$. Thus sending $\lambda \to -\infty$ sends this lower bound on $\mathcal{R}_{\mathrm{MM-REx}}$ to ∞ and hence $\mathcal{R}_{\mathrm{MM-REx}} \to \infty$ as well. □

Proposition 2. *The V-REx risk of predictor f_θ, $\mathcal{R}_{\mathrm{V-REx}}(\theta) \to \infty$ as $\beta \to \infty$ unless $\mathcal{R}^d = \mathcal{R}^e$ for all training domains d, e.*

Proof. Again, let $\epsilon > 0$ be the largest difference in training risks, and let μ be the mean of the training risks. Then there must exist an e such that $|\mathcal{R}_e - \mu| \geq \epsilon/2$. And thus $Var_i(\mathcal{R}_i(\theta)) = \sum_i (\mathcal{R}_i - \mu)^2 \geq (\epsilon/2)^2$, since all other terms in the sum are non-negative. Since $\epsilon > 0$ by assumption, the penalty term is positive and thus $\mathcal{R}_{\mathrm{V-REx}}(\theta) \doteq \sum_i \mathcal{R}_i(\theta) + \beta Var_i(\mathcal{R}_i(\theta))$ goes to infinity as $\beta \to \infty$. □

F.2. Connecting MM-REx to V-REx

F.2.1. Closed form solutions to risk interpolation and minimax-REx

Here, we show that risk interpolation is equivalent to the robust optimization objective of Eqn. 5. Without loss of generality, let $\mathcal{R}_1$ be the largest risk, so $\mathcal{R}_e \leq \mathcal{R}_1$, for all e. Thus we can express $\mathcal{R}_e = \mathcal{R}_1 - d_e$ for some non-negative d_e, with $d_1 = 0 \geq d_e$ for all e. And thus we can write the weighted sum of Eqn. 7 as:

$$\mathcal{R}_{\mathrm{MM}}(\theta) \doteq \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} \lambda_e \mathcal{R}_e(\theta) \tag{12}$$

$$= \max_{\substack{\Sigma_e \lambda_e = 1 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} \lambda_e (\mathcal{R}_1(\theta) - d_e) \tag{13}$$

$$= \mathcal{R}_1(\theta) + \max_{\substack{\Sigma_e \lambda_e = 2 \\ \lambda_e \geq \lambda_{\min}}} \sum_{e=1}^{m} -\lambda_e(d_e) \tag{14}$$

$$\tag{15}$$

Now, since d_e are non-negative, $-d_e$ is non-positive, and the maximal value of this sum is achieved when $\lambda_e = \lambda_{\min}$ for all $e \geq 2$, which also implies that $\lambda_1 = 1 - (m-1)\lambda_{\min}$. This yields the closed form solution provided in Eqn. 7. The special case of Risk Interpolation, where $\lambda_{\min} = 0$, yields Eqn. 5.

F.2.2. Minimax-REx and Mean absolute error REx

In the case of only two training risks, MM-REx is equivalent to using a penalty on the mean absolute error (MAE) between training risks. However, penalizing the pairwise absolute errors is not equivalent when there are $m > 2$ training risks, as we show below. Without loss of generality, assume that $\mathcal{R}_1 < \mathcal{R}_2 < ... < \mathcal{R}_m$. Then $(1/2$ of) the $\mathcal{R}_{\mathrm{MAE}}$ penalty term is:

$$\sum_i \sum_{j \le i} (\mathcal{R}_i - \mathcal{R}_j) = m\mathcal{R}_m - \sum_{j \le m} \mathcal{R}_j + (m-1)\mathcal{R}_{m-1} - \sum_{j \le m-1} \mathcal{R}_j \dots \tag{16}$$

$$= \sum_j j\mathcal{R}_j - \sum_j \sum_{i \le j} \mathcal{R}_i \tag{17}$$

$$= \sum_j j\mathcal{R}_j - \sum_j (m - j + 1)\mathcal{R}_j \tag{18}$$

$$= \sum_j (2j - m - 1)\mathcal{R}_j \tag{19}$$

For $m = 2$, we have $1/2\mathcal{R}_{\text{MAE}} = (2*1-2-1)\mathcal{R}_1 + (2*2-2-1)\mathcal{R}_2 = \mathcal{R}_2 - \mathcal{R}_1$. Now, adding this penalty term with some coefficient β_{MAE} to the ERM term yields:

$$\mathcal{R}_{\text{MAE}} \doteq \mathcal{R}_1 + \mathcal{R}_2 + \beta_{\text{MAE}}(\mathcal{R}_2 - \mathcal{R}_1) = (1 - \beta_{\text{MAE}})\mathcal{R}_1 + (1 + \beta_{\text{MAE}})\mathcal{R}_2 \tag{20}$$

$$\tag{21}$$

We wish to show that this is equal to $\mathcal{R}_{\text{MM}}$ for an appropriate choice of learning rate γ_{MAE} and hyperparameter β_{MAE}. Still assuming that $\mathcal{R}_1 < \mathcal{R}_2$, we have that:

$$\mathcal{R}_{\text{MM}} \doteq (1 - \lambda_{\min})\mathcal{R}_2 + \lambda_{\min}\mathcal{R}_1 \tag{22}$$

Choosing $\gamma_{\text{MAE}} = 1/2\gamma_{\text{MM}}$ is equivalent to multiplying $\mathcal{R}_{\text{MM}}$ by 2, yielding:

$$2\mathcal{R}_{\text{MM}} \doteq 2(1 - \lambda_{\min})\mathcal{R}_2 + 2\lambda_{\min}\mathcal{R}_1 \tag{23}$$

Now, in order for $\mathcal{R}_{\text{MAE}} = 2\mathcal{R}_{\text{MM}}$, we need that:

$$2 - 2\lambda_{\min} = 1 + \beta_{\text{MAE}} \tag{24}$$

$$2\lambda_{\min} = 1 - \beta_{\text{MAE}} \tag{25}$$

$$\tag{26}$$

And this holds whenever $\beta_{\text{MAE}} = 1 - 2\lambda_{\min}$. When $m > 2$, however, these are not equivalent, since R_{MM} puts equal weight on all but the highest risk, whereas $\mathcal{R}_{\text{MAE}}$ assigns a different weight to each risk.

F.2.3. Penalizing pairwise mean squared error (MSE) yields V-REx

The V-REx penalty (Eqn. 8) is equivalent to the average pairwise mean squared error between all training risks (up to a constant factor of 2). Recall that $\mathcal{R}_i$ denotes the risk on domain i. We have:

$$\frac{1}{2n^2} \sum_i \sum_j (\mathcal{R}_i - \mathcal{R}_j)^2 = \frac{1}{2n^2} \sum_i \sum_j \left(\mathcal{R}_i^2 + \mathcal{R}_j^2 - 2\mathcal{R}_i\mathcal{R}_j\right) \tag{27}$$

$$= \frac{1}{2n} \sum_i \mathcal{R}_i^2 + \frac{1}{2n} \sum_j \mathcal{R}_j^2 - \frac{1}{n^2} \sum_i \sum_j \mathcal{R}_i\mathcal{R}_j \tag{28}$$

$$= \frac{1}{n} \sum_i \mathcal{R}_i^2 - \left(\frac{1}{n} \sum_i \mathcal{R}_i\right)^2 \tag{29}$$

$$= \operatorname{Var}(\mathcal{R})\,. \tag{30}$$

G. Further results and details for experiments mentioned in main text

G.1. CMNIST with covariate shift

Here we present the following additional results:

1. Figure 1 of the main text with additional results using MM-REx, see G.1. These results used the “default” parameters from the code of Arjovsky et al. (2019).

2. A plot with results on these same tasks after performing a random search over hyperparameter values similar to that performed by Arjovsky et al. (2019).

3. A plot with the percentage of the randomly sampled hyperparameter combinations that have satisfactory ($> 50\%$) accuracy, which we count as “success” since this is better than random chance performance.

These results show that REx is able to handle greater covariate shift than IRM, given appropriate hyperparameters. Furthermore, when appropriately tuned, REx can outperform IRM in situations with covariate shift. The lower success rate of REx for high values of p is because it produces degenerate results (where training accuracy is less than test accuracy) more often.

The hyperparameter search consisted of a uniformly random search of 340 samples over the following intervals of the hyperparameters:

1. HiddenDim = [2**7, 2**12]

2. L2RegularizerWeight = [10**-2, 10**-4]

3. Lr = [10**-2.8, 10**-4.3]

4. PenaltyAnnealIters = [50, 250]

5. PenaltyWeight = [10**2, 10**6]

6. Steps = [201, 601]

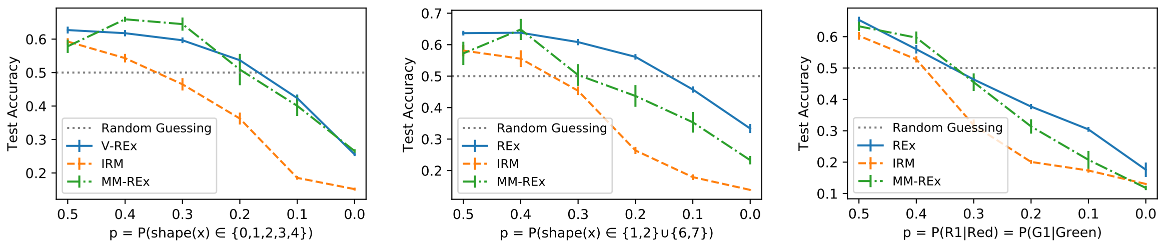

Figure 10. This is Figure 5 of main text with additional results using MM-REx. For each covariate shift variant (class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text) of CMNIST, the standard error (the vertical bars in plots) is higher for MM-REx than for V-REx.

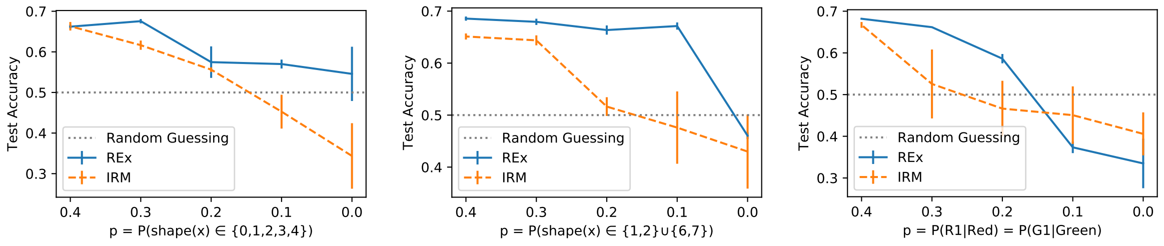

Figure 11. This is Figure 5 of main text (class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text), but with hyperparameters of REx and IRM each tuned to perform as well as possible for each value of p for each covariate shift type.

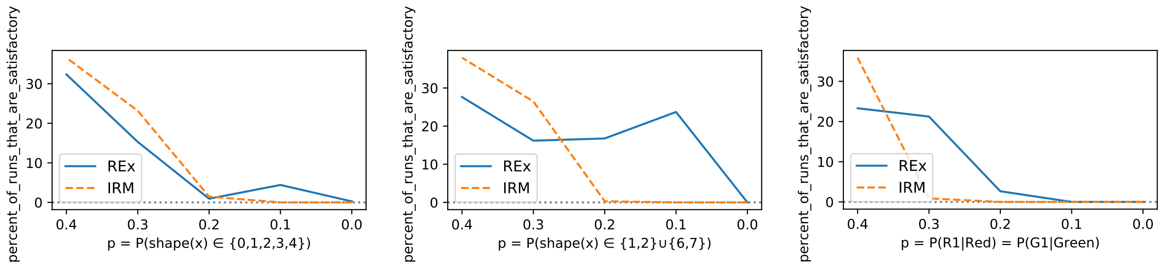

Figure 12. This also corresponds to class imbalance, digit imbalance, and color imbalance from left to right as described in "CMNIST with covariate shift" subsubsection of Section 4.1 in main text; but now the y-axis refers to what percentage of the randomly sampled hyperparameter combinations we deemed to to be satisfactory. We define satisfactory as simultaneously being better than random guessing and having train accuracy greater than test accuracy. For p less than .5, a larger percentage of hyperparameter combinations are often satisfactory for REx than for IRM; for p greater than .5, a larger percentage of hyperparameter combinations are often satisfactory for IRM than for REx because train accuracy is greater than test accuracy for more hyperparameter combinations for IRM. We stipulate that train accuracy must be greater than test accuracy because test accuracy being greater than train accuracy usually means the model has learned a degenerate prediction rule such as "not color".

G.2. SEMs from "Invariant Risk Minimization"

Here we present experiments on the (linear) structural equation model (SEM) tasks introduced by Arjovsky et al. (2019). Arjovsky et al. (2019) construct several varieties of SCM where the task is to predict targets Y from inputs X_1, X_2, where X_1 are (non-anti-causal) causes of Y, and X_2 are (anti-causal) effects of Y. We refer the reader to Section 5.1 and Figure 3 of Arjovsky et al. (2019) for more details. We use the same experimental settings as Arjovsky et al. (2019) (except we only run 7 trials), and report results in Table 5.

These experiments include several variants of a simple SEM, given by:

$$\begin{aligned} X_1 &= N_1 \\ Y &= W_{1\to Y} X_1 + N_Y \\ X_2 &= W_{Y\to 2} Y + N_2 \end{aligned}$$

Where N_1, N_Y, N_2 are all sampled i.i.d. from normal distributions. The variance of these distributions may vary across domains.

While REx achieves good performance in the **domain-homoskedastic** case, it performs poorly in the **domain-heteroskedastic** case, where the amount of intrinsic noise, σ_y^2 in the target changes across domains.[15] Intuitively, this is because the irreducible error varies across domains in these tasks, meaning that the risk will be larger on some domains than others, even if the model's predictions match the expectation $\mathbb{E}(Y|Pa(Y))$. We tried using a "baseline" (see Eqn. 5) of $r_e = Var(Y_e)$ (Meinshausen et al., 2015) to account for the different noise levels in Y, but this did not work.

We include a mathematical analysis of the simple SCM given above in order to better understand why REx succeeds in the domain-homoskedastic, but not the domain-heteroskedastic case. Assuming that Y, X_1, X_2 are scalars, this SCM becomes

$$\begin{aligned} X_1 &= N_1 \\ Y &= w_{1\to y} N_1 + N_Y \\ X_2 &= w_{y\to 2} w_{1\to y} N_1 + w_{y\to 2} N_Y + N_2 \end{aligned}$$

We consider learning a model $\hat{Y} = \alpha X_1 + \beta X_2$. Then the residual is:

$$\hat{Y} - Y = (\alpha + w_{1\to y}(\beta w_{y\to 2} - 1))N_1 + (\beta w_{y\to 2} - 1)N_Y + \beta N_2$$

Since all random variables have zero mean, the MSE loss is the variance of the residual. Using the fact that the noise N_1, N_Y, N_2 are independent, this equals:

$$\mathbb{E}[(\hat{Y} - Y)^2] = (\alpha + w_{1\to y}(\beta w_{y\to 2} - 1))^2\sigma_1^2 + (\beta w_{y\to 2} - 1)^2\sigma_Y^2 + \beta^2\sigma_2^2$$

[15] See Footnote 11.

	FOU(c)	FOU(nc)	FOS(c)	FOS(nc)
IRM	0.001±0.000	0.001±0.000	0.001±0.000	0.000±0.000
REx, $r_e = 0$	0.001±0.000	0.008±0.002	0.007±0.002	0.000±0.000
REx, $r_e = \mathbb{V}(Y_e)$	0.816±0.149	1.417±0.442	0.919±0.091	0.000±0.000
	POU(c)	POU(nc)	POS(c)	POS(nc)
IRM	0.004±0.001	0.006±0.003	0.002±0.000	0.000±0.000
REx, $r_e = 0$	0.004±0.001	0.004±0.001	0.002±0.000	0.000±0.000
REx, $r_e = \mathbb{V}(Y_e)$	0.915±0.055	1.113±0.085	0.937±0.090	0.000±0.000
	FEU(c)	FEU(nc)	FES(c)	FES(nc)
IRM	0.0053±0.0015	0.1025±0.0173	0.0393±0.0054	0.0000±0.0000
REx, $r_e = 0$	0.0390±0.0089	19.1518±3.3012	7.7646±1.1865	0.0000±0.0000
REx, $r_e = \mathbb{V}(Y_e)$	0.7713±0.1402	1.0358±0.1214	0.8603±0.0233	0.0000±0.0000
	PEU(c)	PEU(nc)	PES(c)	PES(nc)
IRM	0.0102±0.0029	0.0991±0.0216	0.0510±0.0049	0.0000±0.0000
REx, $r_e = 0$	0.0784±0.0211	46.7235±11.7409	8.3640±2.6108	0.0000±0.0000
REx, $r_e = \mathbb{V}(Y_e)$	1.0597±0.0829	0.9946±0.0487	1.0252±0.0819	0.0000±0.0000

Table 5. Average mean-squared error between true and estimated weights on causal (X_1) and non-causal (X_2) variables. **Top 2:** When the level of noise in the anti-causal features varies across domains, REx performs well (FOU, FOS, POU, POS). **Bottom 2:** When the level of noise in the targets varies instead, REx performs poorly (FEU, FES, PEU, PES). Using the baselines $r_e = \mathbb{V}(Y)$ does not solve the problem, and indeed, hurts performance on the homoskedastic domains.

Thus when (only) σ_2 changes, the only way to keep the loss unchanged is to set the coefficient in front of σ_2 to 0, meaning $\beta = 0$. By minimizing the loss, we then recover $\alpha = w_{1\to y}$; i.e. in the domain-homoskedastic setting, the loss equality constraint of REx yields the causal model. On the other hand, if (only) σ_Y changes, then REx enforces $\beta = 1/w_{y\to 2}$, which then induces $\alpha = 0$, recovering the *anti*causal model.

While REx (like ICP (Peters et al., 2016)) assumes the mechanism for Y is fixed across domains (meaning $P(Y|Pa(Y))$ is independent of the domain, e), IRM makes the somewhat weaker assumption that $\mathbb{E}(Y|Pa(Y))$ is independent of domain. While it is plausible that an appropriately designed variant of REx could work under this weaker assumption, we believe forbidding interventions on Y is not overly restrictive, and such an extension for future work.

G.3. Reinforcement Learning Experiments

Here we provide details and further results on the experiments in Section 4.1. We take tasks from the Deepmind Control Suite (Tassa et al., 2018) and modify the original state, $\mathbf{s}$, to produce observation, $\mathbf{o} = (\mathbf{s} + \epsilon, \eta\mathbf{s}')$ including noise ϵ and spurious features $\eta\mathbf{s}'$, where $\mathbf{s}'$ contains 1 or 2 dimensions of $\mathbf{s}$. The scaling factor takes values $\eta = 1/2/3$ for the two training and test domains, respectively. The agent takes $\mathbf{o}$ as input and learns a representation using Soft Actor-Critic (Haarnoja et al., 2018) and an auxiliary reward predictor, which is trained to predict the next 3 rewards conditioned on the next 3 actions. Since the spurious features are copied from the state before the noise is added, they are more informative for the reward prediction task, but they do not have an invariant relationship with the reward because of the domain-dependent η.

The hyperparameters used for training Soft Actor-Critic can be found in Table 6. We used `cartpole_swingup` as a development task to tune the hyperparameters of penalty weight (chosen from $[0.01, 0.1, 1, 10]$) and number of iterations before the penalty is turned up (chosen from $[5000, 10000, 20000]$), both for REx and IRM. The plots with the hyperparameter sweep are in Figure 13.

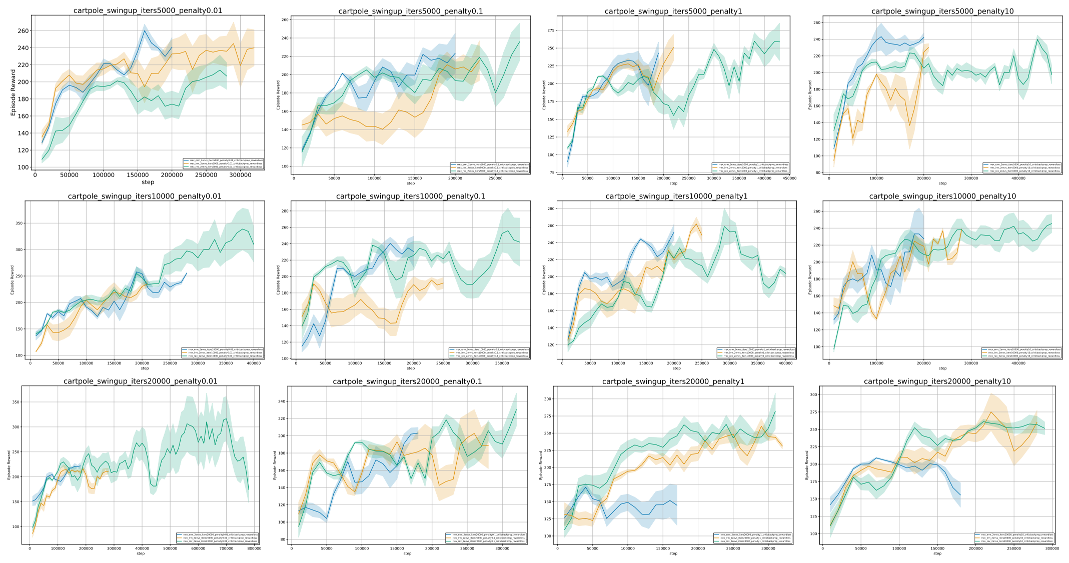

Figure 13. Hyperparameter sweep for IRM and REx on `cartpole_swingup`. Green, blue, and orange curves correspond to REx, ERM, and IRM, respectively. The subfigure titles state the penalty strength ("penalty") and after how many iterations the penalty strength was increased ("iters"). We chose a penalty factor of 1 and 10k iterations.

Parameter name	Value
Replay buffer capacity	1000000
Batch size	1024
Discount γ	0.99
Optimizer	Adam
Critic learning rate	10^{-5}
Critic target update frequency	2
Critic Q-function soft-update rate τ_{Q}	0.005
Critic encoder soft-update rate τ_{enc}	0.005
Actor learning rate	10^{-5}
Actor update frequency	2
Actor log stddev bounds	$[-5, 2]$
Encoder learning rate	10^{-5}
Decoder learning rate	10^{-5}
Decoder weight decay	10^{-7}
L1 regularization weight	10^{-5}
Temperature learning rate	10^{-4}
Temperature Adam's β_1	0.9
Init temperature	0.1

Table 6. A complete overview of hyperparameters used for reinforcement learning experiments.

H. Experiments not mentioned in main text

We include several other experiments which do not contribute directly to the core message of our paper. Here is a summary of the take-aways from these experiments:

1. Our experiments in the CMNIST domain suggest that the IRM/V-REx penalty terms should be amplified exactly when the model starts overfitting training distributions.

2. Our financial indicators experiments suggest that IRM and REx often perform remarkably similarly in practice.

H.1. A possible approach to scheduling IRM/REx penalties

We've found that REx and IRM are quite sensitive to the choice of hyperparameters. In particular, hyperparameters controlling the scheduling of the IRM/V-REx penalty terms are of critical importance. For the best performance, the penalty should be increased the relative weight of the penalty term after approximately 100 epochs of training (using a so-called "waterfall" schedule (Desjardins et al., 2015)). See Figure 14(b) for a comparison. We also tried an exponential decay schedule instead of the waterfall and found the results (not reported) were significantly worse, although still above 50% accuracy.

Given the methodological constraints of out-of-distribution generalization mentioned in (Gulrajani & Lopez-Paz, 2020), this could be a significant practical issue for applying these algorithms. We aim to address this limitation by providing a guideline for when to increase the penalty weight, based only on the training domains. We hypothesize that successful learning of causal features using REx or IRM should proceed in two stages:

1. In the first stage, predictive features are learned.

2. In the second stage, causal 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.

The experiments presented in this section provide *observational* evidence consistent with this hypothesis. However, since the hypothesis was developed by observing patterns in the CMNIST training runs, it requires further experimental validation on a different task, which we leave for future work.

H.1.1. Results and Interpretation

In Figure 14, we demonstrate that the optimal point to apply the waterfall in the CMNIST task 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 anti-causal and causal 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 15.

H.2. Domain Generalization: VLCS and PACS

Here we provide earlier experiments on the VLCS and PACS dataset. We removed these experiments from the main text of our paper in favor of the more complete DomainBed results.

To test whether REx provides a benefit on more realistic domain generalization tasks, we compared 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.

Since every domain in PACS is used as a test set when training on the other three domains, it is not possible to perform a methodologically sound evaluation on PACS after examining results on *any* of the data. Thus to avoid performing any tuning on test distributions, we use VLCS to tune hyperparameters and then apply these exact same settings to PACS and report the final average over 10 runs on each domain.

We use the same architecture, training procedure and data augmentation strategy as the (formerly) state-of-the-art Jigsaw Puzzle approach (Carlucci et al., 2019) (except with IRM or V-REx intead 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. Results on PACS are in Table 8. On PACS we found that REx outperforms IRM and IRM outperforms ERM on average, while all are worse than the state-of-the-art Jigsaw method.

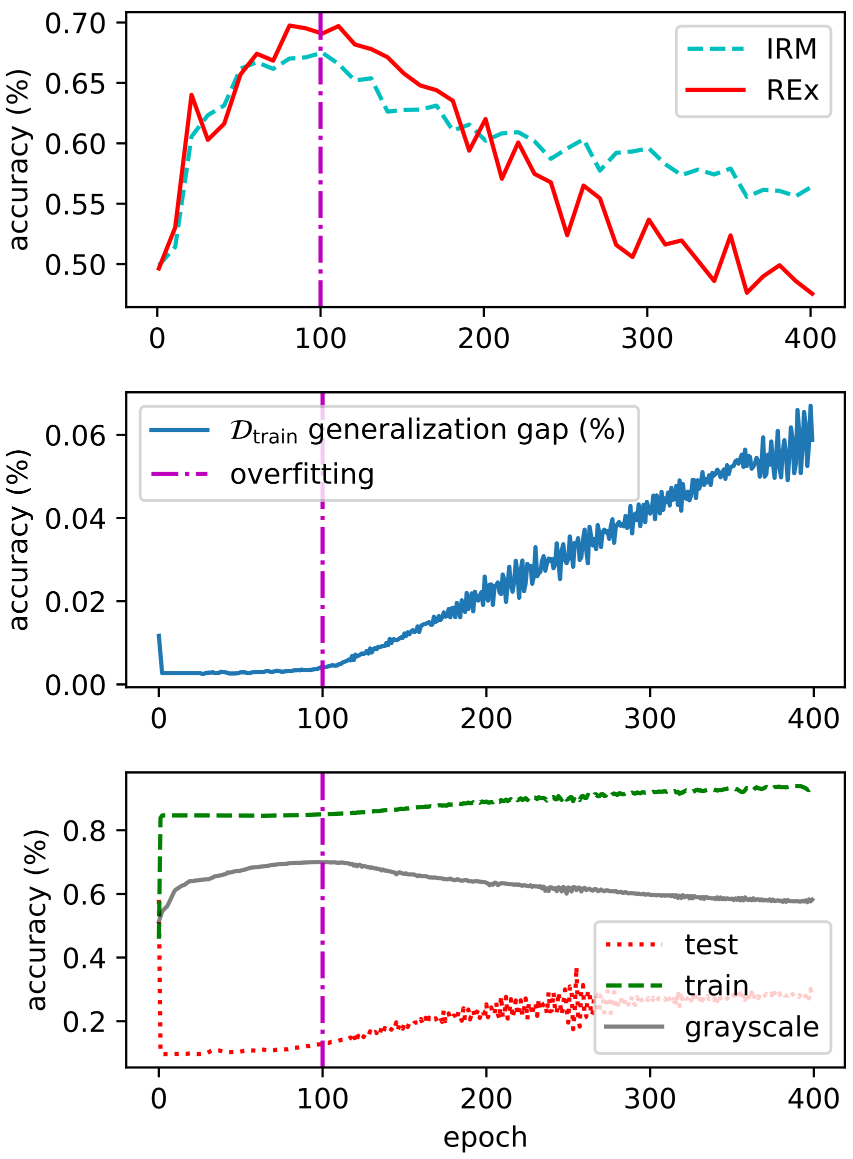

Figure 14. Stability penalties should be applied around when traditional overfitting begins, to ensure that the model has learned predictive features, and that penalties still give meaningful training signals. **Top:** Test accuracy as a function of epoch at which penalty term weight is increased (learning 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. **Bottom:** 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.

We use all hyperparameters from the original Jigsaw codebase.[16] We use Imagenet pre-trained AlexNet features and chose batch-size, learning rate, as well as penalty weights based on performance on the VLCS dataset where test performance on the holdout domain was used for the set of parameters producing the highest validation accuracy. The best performing

[16] https://github.com/fmcarlucci/JigenDG

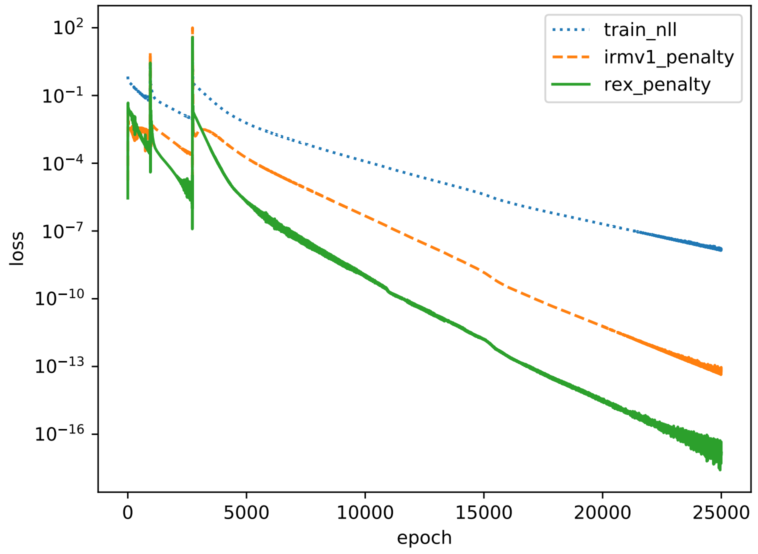

Figure 15. Given sufficient training time, empirical risk minimization (ERM) minimizes both REx and IRMv1 penalty terms on Colored MNIST (*without* including either term in the loss function). This is because the model (a deep network) has sufficient capacity to fit the training sets almost perfectly. This prevents these penalties from having the intended effect, once the model has started to overfit. The y-axis is in log-scale.

parameters on VLCS were then applied to the PACS dataset without further changes. We searched over batch-sizes in $\{128, 384\}$, over penalty strengths in $\{0.0001, 0.001, 0.01, 0.1, 1, 10\}$, learning rates in $\{0.001, 0.01\}$ and used average performance over all 4 VLCS domains to pick the best performing hyperparameters. Table 7 shows results on VLCS with the best performing hyperparameters.

The final parameters for all methods on PACS were a batch size of 384 with 30 epochs of training with Adam, using a learning rate of 0.001, and multiplying it by 0.1 after 24 epochs (this step schedule was taken from the Jigsaw repo).The penalty weight chosen for Jigsaw was 0.9; for IRM and REx it was 0.1.We used the same data-augmentation pipeline as the original Jigsaw code for ERM, IRM, Jigsaw and REx to allow for a fair comparison.

VLCS	CALTECH	SUN	PASCAL	LABELME	Average
REx (ours)	**~~96.72~~**	~~63.68~~	**~~72.41~~**	~~60.40~~	**~~73.30~~**
IRM	~~95.99~~	~~62.85~~	~~71.71~~	~~59.61~~	~~72.54~~
ERM	~~94.76~~	~~61.92~~	~~69.03~~	**~~60.55~~**	~~71.56~~
Jigsaw (SOTA)	~~96.46~~	**~~63.84~~**	~~70.49~~	~~60.06~~	~~72.71~~

Table 7. Accuracy (percent) of different methods on the VLCS task. Results are test accuracy at the time of the highest validation accuracy, averaged over 10 runs. On VLCS REx outperforms all other methods. Numbers are shown in strike-through because we selected our hyperparameters based on highest test set performance; the goal of this experiment was to find suitable hyperparameters for the PACS experiment.

H.3. Financial indicators

We find that IRM and REx seem to perform similarly across different splits of the data in a prediction task using financial data. 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 (see Appendix for dataset details.) We consider each year a different domain, and create 20 different tasks by selecting all possible combinations of domains where three domains represent the training sets, one domain the validation set, and

PACS	Art Painting	Cartoon	Sketch	Photo	Average
REx (ours)	66.27±0.46	68.8±0.28	59.57±0.78	89.60±0.12	71.07
IRM	66.46±0.31	68.60±0.40	58.66±0.73	89.94±0.13	70.91
ERM	66.01±0.22	68.62±0.36	58.38±0.60	89.40±0.18	70.60
Jigsaw (SOTA)	66.96±0.39	66.67±0.41	61.27±0.73	89.54±0.19	71.11

Table 8. 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 on average, and performs similar to IRM and Jigsaw (the state-of-the-art).

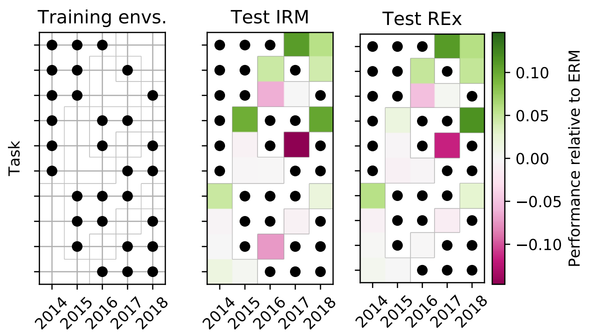

Figure 16. Financial indicators tasks. The left panel indicates the set of training domains; the middle and right panels show the test accuracy on the respective domains relative to ERM (a black dot corresponds to a training domain; a colored patch indicates the test accuracy on the respective domain.)

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. 16 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. While performance on specific tasks differs significantly between ERM and IRM/REx, performance averaged over tasks is not significantly different.

H.3.1. Experiment Details

We use `v1` of the dataset published on [17] and prepare the data as described in.[18] We further remove all the variables that are not shared across all 5 years, leaving us with 37 features, and whiten the data through centering and normalizing by the standard deviation.

On each subtask, we train an MLP with two hidden layers of size 128 with tanh activations and dropout (p=0.5) after each layer. We optimize the binary cross-entropy loss using Adam (learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$), and an L2 penalty (weight 0.001). In the IRM/REx experiments, the respective penalty is added to the loss ($\beta = 1$) and the original loss is scaled by a factor 10^{-4} after 1000 iterations. Experiments are run for a maximum of 9000 training iterations with early stopping based on the validation performance. All results are averaged over 3 trials. The overall performance of the different models, averaged over all tasks, is summarized in Tab. 9. The difference in average performance between ERM, IRM, and REx is not statistically significant, as the error bars are very large.

I. Overview of other topics related to OOD generalization

Domain adaptation (Ben-David et al., 2010a) shares the goal of generalizing to new distributions at test time, but allows some access to the test distribution. A common approach is to make different domains have a similar distribution of features (Pan et al., 2010). A popular deep learning method for doing so is Adversarial Domain Adaptation (ADA) (Ganin et al., 2016; Tzeng et al., 2017; Long et al., 2018; Li et al., 2018), which seeks a "invariant representation" of the inputs, i.e. one

[17] `https://www.kaggle.com/cnic92/200-financial-indicators-of-us-stocks-20142018`
[18] `https://www.kaggle.com/cnic92/explore-and-clean-financial-indicators-dataset`

	Overall accuracy	Min acc.	Max acc.
ERM	54.6 ± 4.6	47.6	66.2
IRM	55.3 ± 5.9	45.9	67.5
REx	$\mathbf{55.5} \pm \mathbf{6.0}$	47.2	**68.0**

Table 9. Test accuracy of models trained on the financial domain dataset, averaged over all 20 tasks, as well as min./max. accuracy across the tasks.

whose distribution is domain-independent. Recent works have identified fundamental shortcomings with this approach, however (Zhao et al., 2019; Johansson et al., 2019; Arjovsky et al., 2019; Wu et al., 2020).

Complementary to the goal of domain generalization is **out-of-distribution detection** (Hendrycks & Gimpel, 2016; Hendrycks et al., 2018), where the goal is to recognize examples as belonging to a new domain. Three common deep learning techniques that can improve OOD generalization are **adversarial training** (Goodfellow et al., 2014; Hendrycks & Dietterich, 2019), **self-supervised learning** (van den Oord et al., 2018; Hjelm et al., 2018; Hendrycks et al., 2019b; Albuquerque et al., 2020) and **data augmentation** (Krizhevsky et al., 2012; Zhang et al., 2017; Cubuk et al., 2018; Shorten & Khoshgoftaar, 2019; Hendrycks et al., 2019a; Carlucci et al., 2019). These methods can also been combined effectively in various ways (Tian et al., 2019; Bachman et al., 2019; Gowal et al., 2019). Data augmentation and self-supervised learning methods typically use prior knowledge such as 2D image structure. Several recent works also use prior knowledge to design augmentation strategies for invariance to superficial features that may be spuriously correlated with labels in object recognition tasks (He et al., 2019; Wang et al., 2019; Gowal et al., 2019; Ilse et al., 2020). In contrast, REx can discover which features have invariant relationships with the label without such prior knowledge.