Rich Feature Construction for the Optimization-Generalization Dilemma

Jianyu Zhang [1] **David Lopez-Paz** [2] **Léon Bottou** [3 1]

Abstract

There often is a dilemma between ease of optimization and robust out-of-distribution (OoD) generalization. For instance, many OoD methods rely on penalty terms whose optimization is challenging. They are either too strong to optimize reliably or too weak to achieve their goals.

We propose to initialize the networks with a rich representation containing a palette of potentially useful features, ready to be used by even simple models. On the one hand, a rich representation provides a good initialization for the optimizer. On the other hand, it also provides an inductive bias that helps OoD generalization. Such a representation is constructed with the Rich Feature Construction (RFC) algorithm, also called the *Bonsai* algorithm,[1] which consists of a succession of training episodes. During *discovery episodes*, we craft a multi-objective optimization criterion and its associated datasets in a manner that prevents the network from using the features constructed in the previous iterations. During *synthesis episodes*, we use knowledge distillation to force the network to simultaneously represent all the previously discovered features.

Initializing the networks with Bonsai representations consistently helps six OoD methods achieve top performance on COLOREDMNIST benchmark (Arjovsky et al., 2020). The same technique substantially outperforms comparable results on the Wilds CAMELYON17 task (Koh et al., 2021), eliminates the high result variance that plagues other methods, and makes hyperparameter tuning and model selection more reliable.

1. Introduction

The interplay of optimization and generalization plays a crucial role in deep learning. It also changes nature when we focus on out-of-distribution (OoD) generalization, that is when training and testing sets are no longer assumed to follow the same distribution.

Simple optimization algorithms are surprisingly able to find uneventful descent paths in the non-convex cost landscape of deep learning networks (Gu et al., 2021; Sagun et al., 2017). However they also tend to construct features that capture spurious correlations (Szegedy et al., 2014; Beery et al., 2018; Arjovsky et al., 2020; Ilyas et al., 2019). Several recent papers propose to work around this problem by leveraging multiple training sets that illustrate the possible changes in distribution. The training algorithm must then satisfy certain constraints across training sets, usually enforced with additional penalty terms (Arjovsky et al., 2020; Koyama & Yamaguchi, 2020; Krueger et al., 2020; Pezeshki et al., 2020; Shi et al., 2021; Rame et al., 2021; Wald et al., 2021). The resulting optimization problem often turns substantially more challenging than the simple empirical risk minimization (ERM). In practice, it is often necessary to schedule the penalty hyper-parameters in a manner that weakens them so much that they no longer enforce the intended constraints. As a result, when one initializes such a method with the correct solution, the training process deviates from the constraint set and finds inferior solutions (Figure 2).

We propose in this paper to work around the difficulties of the optimization problem by first obtaining a representation of the input patterns that contains a broad diversity of potentially useful features. Both the ERM and OoD methods can then easily pick the most useful features, according to the chosen training cost and constraints, in a much easier way. The Rich Feature Construction (RFC) algorithm, informally called the Bonsai algorithm, (Section 4) consists of a succession of training episodes. During the *discovery episodes*, we craft a multi-objective optimization criterion that prevents the network from using the features constructed in the previ-

[1]New York University, New York, NY, USA. [2]Facebook AI Research, Paris, France. [3]Facebook AI Research, New York, NY, USA.. Correspondence to: Jianyu Zhang <jianyu@nyu.edu>.

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

[1]Bonsai, also known as *penjing* (tray planting), refers to the art of growing small trees in trays using obsessive trimming techniques to impede their growth and produce miniature versions of real-life trees. Likewise, the RFC algorithm impedes the learning process in order to obtain diverse representations.

ous steps. During the *synthesis episodes*, we force the final representation to simultaneously represent all the previously identified features.

On the common out-of-distribution (OoD) COLOREDMNIST task (Arjovsky et al., 2020), we show that initializing networks with Bonsai representations consistently helps six state-of-the-art OoD learning algorithms learn the robust feature and disregard the spurious one. The same method also performs well on a modified task, REVERSECOLOREDMNIST, in which the robust feature is made more predictive than the spurious feature. Such a modification breaks all methods that aim for the 2nd easiest-to-find features (Nam et al., 2020; Liu et al., 2021; Bao et al., 2021). Finally, we evaluate Bonsai initialization on the real-world CAMELYON17 task (Koh et al., 2021) and show that it not only helps OoD and ERM methods match or exceed the best published results, but also facilitates the hyper-parameter tuning and model selection.[2]

2. Related Work

2.1. Leveraging multiple training environments

In order to achieve a good performance on testing data that follows a different distribution from the training data, many OoD methods assume access to multiple training sets, or environments, whose different distributions illustrate a range of potential distribution changes. One possible direction consists of learning a representation such that the optimal classifier built on top is the same for all training environments: IRMv1/IRM (Arjovsky et al., 2020), MAML-IRM (Bae et al., 2021), CLOvE (Wald et al., 2021). Another line of work introduces gradient alignment constraints across training environments using dot-product (Fish (Shi et al., 2021)), squared distance of gradients (IGA (Koyama & Yamaguchi, 2020)), or squared distance of gradients variance (Fishr (Rame et al., 2021)). Methods such as vREx (Krueger et al., 2020) and GroupDRO (Sagawa et al., 2019) aim at finding a solution that performs equally well across training environments.

2.2. Facilitating the optimization

In contrast, the SD method (Pezeshki et al., 2020) relies on a single training set but fights the "gradient starvation" phenomenon that prevents the training algorithm from finding robust features even though they are assumed more predictive on the training set than easier-to-find spurious features.

[2]Code for replicating these experiments is publicly available at `https://github.com/TjuJianyu/RFC/`.

2.3. Aiming for the second easiest representation

Closely related to the RFC *discovery episodes*, several methods seek the *second easiest-to-find representation*, either by reweighing the dataset (Liu et al., 2021; Nam et al., 2020) or with distribution robust optimization, (Bao et al., 2021; Ahmed et al., 2020; Creager et al., 2021). The main drawback of these methods is the assumption that the second-easiest representation is the correct one. This happens to be true in benchmarks such as COLOREDMNIST which are designed to frustrate ERM. However, these methods fail on simpler tasks that do not follow the assumption (Section 5.2).

2.4. Making use of diverse features

Closely related to the RFC *synthesis episodes*, other methods attempt to steer the training process towards constructing a diversity of features: RSC (Huang et al., 2020) masks out features associated with large gradients to force the last layer to pick additional features; DiverseModel (Teney et al., 2021) constructs multiple classifiers on top of a given representation (e.g. ImageNet pre-trained) with a penalty that minimizes the alignment of gradients across classifiers. The drawback of these approaches is that they work best when starting from an existing portfolio of diverse features.

3. The optimization-generalization dilemma

This section presents experiments that illustrate the optimization-generalization dilemma that plagues OoD methods. All these experiments are carried out on the COLORMNIST task (Arjovsky et al., 2020). In this task, the relation between the robust feature (the digit class) and output label is invariant in all training and testing environments. In contrast, although the spurious feature (the digit color) is more predictive on the training environments, its relation with the output labels is not invariant across environments. We report results on a variety of published OoD algorithms: IRMv1 (Arjovsky et al., 2020), Fish (Shi et al., 2021), IGA (Koyama & Yamaguchi, 2020), vREx (Krueger et al., 2020), Spectral Decoupling (SD) (Pezeshki et al., 2020), Fishr (Rame et al., 2021), RSC (Huang et al., 2020), LfF (Nam et al., 2020), and CLOvE (Wald et al., 2021). We do not report results on MAML-IRM (Bae et al., 2021) because it is equivalent to Fish+vREx, and we do not report results on GroupDRO (Sagawa et al., 2019) because it performs like vREx (see Appendix A and B for details).

3.1. OoD penalties make the optimization challenging

Because their optimization is difficult, most authors recommend to pre-train the network with ERM before applying their OoD method. Figure 1 shows the final OoD test performance of models trained with each method as a function of

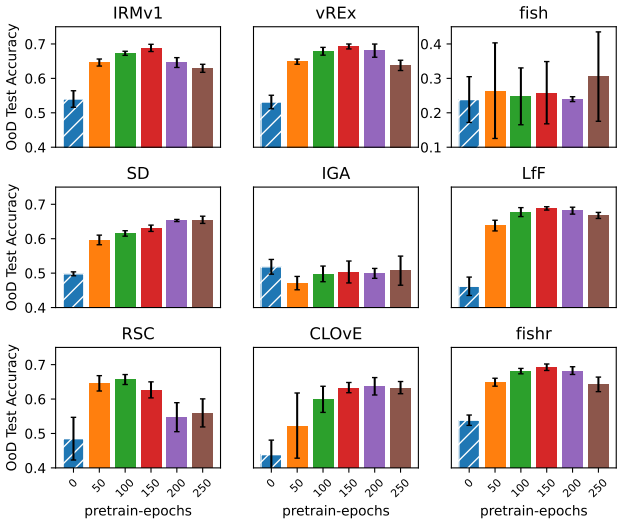

Figure 1. Test performance of nine penalized OoD methods as a function of the number of epochs used to pre-train the neural network with ERM. The final OoD testing performance is very dependent on choosing the right number of pretraining epochs, illustrating the challenges of these optimization problems.

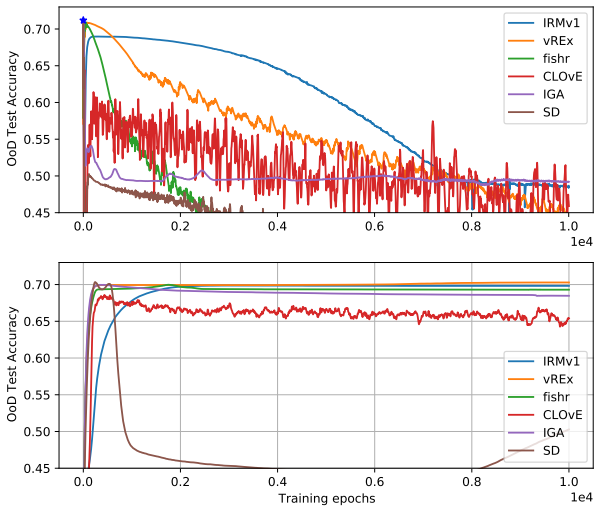

Figure 2. Test performance of OoD methods as a function of training epochs. Top: Six OoD methods are trained from a 'perfect' initialization where only the robust feature is well learned. The blue star indicates the initial test accuracy. Bottom: The OoD methods are trained from the proposed (frozen) Bonsai representation.

the number $n_p \in \{0, 50, 100, 150, 200, 250\}$ of ERM pretraining epochs. During the execution of the OoD algorithm, we choose one of five penalty weights and select the best early-stopping epoch by directly peeking at the OoD test performance. All other hyper-parameters are copied from the COLORMNIST task (Arjovsky et al., 2020). Appendix D discusses these experiments with further details.

Figure 1 shows that optimizing from a random initialization (blue bars, 0 pretraining epochs) fails for all nine algorithms and all five penalty weights. Although pretraining with ERM helps, the final performance of the competitive algorithms depends on the number of pretraining epochs in rather inconsistent. Too much pretraining can cause performance drops in excess of 20%. Even when one guesses the right amount of pretraining, the final performance comes short of the oracle performance (0.721 ± 0.002) achieved by a network that is trained only on the robust feature.

We also showcase the optimization difficulty of several OoD methods from a loss landscape's view on a low-dimensional case. See Appendix C for details.

3.2. OoD penalties do not enforce the constraints

The previous section shows that the penalties introduced by these OoD methods are too strong to allow reliable optimization. We now show that they are also too weak to enforce the constraints they are meant to enforce.

To substantiate this assertion, we initialize a network with the correct solution, that is, the solution obtained by training the network on a variant of the COLORMNIST dataset in which the spurious color feature was removed. In order to keep the network from deviating from the target constraint, we use the largest penalty weight in the search space in each OoD method. We do not report results on the Fish method because it failed to learn the task. We do not report on RSC and LfF because their test accuracy drops too fast.

The top plot in figure 2 shows how the OoD testing performance of six algorithms deviates from the performance of our perfect initialization. This might happen because the chosen constraints have spurious solutions (Kamath et al., 2021) or because the penalty terms are too weak to enforce the target constraints. Instead, the training process pulls the perfect initialization in the direction of the spurious feature (the color) which happens to be more predictive on the training data.

4. Rich Feature Construction

This section presents tools for constructing rich representations. First, we describe a mathematically sound approach to the problem of constructing a rich set of diverse features and we introduce the notions of *discovery* and *synthesis* episodes. Then we show how to use Distributionally Robust Optimization (DRO) to cut on the synthesis episodes. Finally, we present the practical Bonsai algorithm that we use in Section 5.

4.1. Feature discovery

Intuitively, constructing additional features is desirable when using these features increases the system performance on a pertinent subset of examples. Best would of course achieve a large performance increase on large subsets of examples.

For the purposes of this section, let $\Phi_k(x)$ be a large vector containing all previously constructed k features for pattern x. Our first step consists of defining an ensemble $P = \{D^1 \dots D^i \dots\}$ of pertinent subsets D^i of the training set. An effective way to choose a good ensemble of subsets is discussed at the end of this section. Having defined such subsets, we can define costs $C_i(\Phi, w)$ that measure the quality of a feature set Φ measured on subset D^i:

$$C_i(\Phi, w) = \frac{1}{|D^i|} \sum_{(x,y)\in D^i} \ell(y, w^\top \Phi(x))$$

where w represent the weights of a linear layer and $\ell(y, \hat{y})$ is a convex loss. In the context of deep learning, considering a linear layer operating a large feature vector is not an unreasonable way to investigate the effectiveness of a representation (Jacot et al., 2018). We can reweigh the training data in a manner that emphasizes the weaknesses of our current set of features, that is,

$$R_{rw} = \max_{\lambda} \min_{w} \sum_i \lambda_i C_i(\Phi_k, w) \tag{1}$$

where the λ_i coefficients are positive and sum to 1. Let λ_i^* be the pessimal mixture coefficients resulting from optimization problem (1). We can then learn a new set of features that help performance on this pessimal mixture,

$$R'_{rw} = \min_{w,\Phi} \sum_i \lambda_i^* C_i(\Phi, w)\ . \tag{2}$$

The main difference is that we are now training the features, yielding a new feature vectors $\Phi^*(x)$. If R'_{rw} (2) is smaller than R_{rw} (1), then we know that Φ^* contains *new useful features that were not present* in Φ_k. This is the *discovery phase*.

The next step consists in forming new feature vectors $\Phi_{k+1}(x)$ that contain the features present in both Φ_k and Φ^*, a *synthesis phase*. We can then iterate and obtain additional useful and diverse features at each iteration. The synthesis phase can be as simple as a vector concatenation. In the context of deep learning, however, one often has to use distillation, as discussed later in section 4.

The selection of a pertinent ensemble of subsets certainly affects which new features will be constructed at each iteration. In particular, it is desirable to make R_{rw} as high as possible using a minimal number of subsets. This goal can be easily achieved by forming subsets containing examples that were either correctly classified or misclassified by the learning systems constructed by problem (2).

4.2. Using DRO

We now show how a DRO reformulation of this process can cut the intermediate synthesis phase. Because the C_i are convex in w, we can first apply von Neumann's minimax theorem (Simons, 1995, theorem 3) to problem (1) and obtain a DRO problem (Ben-Tal et al., 2009):

$$\begin{aligned} R_{rw} \ &= \ \max_{\lambda} \min_{w} \sum_i \lambda_i C_i(\Phi_k, w) \\ &= \min_{w} \max_{\lambda} \sum_i \lambda_i C_i(\Phi_k, w) \\ &= \min_{w} \max_{i} C_i(\Phi_k, w) \ = \ R_{dro}\ . \end{aligned} \tag{3}$$

The next step is to run this same DRO optimization while also learning the features

$$R'_{dro} = \min_{w,\Phi} \max_{i} C_i(\Phi, w)\ . \tag{4}$$

To understand how quantity R'_{dro} relates to R'_{rw}, we can use the max-min inequality as follows:

$$\begin{aligned} R'_{dro} &= \min_{w,\Phi} \max_{\lambda} \sum_i \lambda_i C_i(\Phi, w) \\ &\geq \max_{\lambda} \min_{w,\Phi} \sum_i \lambda_i C_i(\Phi, w) \\ &\geq \min_{w,\Phi} \sum_i \lambda_i^* C_i(\Phi, w) = R'_{rw}\ . \end{aligned}$$

In other words, if R'_{dro} is smaller than R_{dro}, then R'_{rw} is smaller than $R_{rw} = R_{dro}$, and the new feature vector Φ contains new and useful features. The advantage of this approach is that problem (4) does not involve mixture coefficients λ^*. Therefore there is no need to solve (3) or (1), and no need for a synthesis phase at each iteration. The synthesis phase is only needed to construct the final rich representation after the last iteration.

4.3. The practical Bonsai algorithm

We now describe a practical algorithm that implements the ideas discussed in the previous subsection in a manner that is usable with ordinary deep networks. The workhorse of this algorithm is the Robust Empirical Risk Minimisation (RERM) algorithm (Algorithm 1) which takes an ensemble of datasets D^k representing multiple distributions and seeks neural network weights that simultaneously yields small errors for all these distributions. RERM is in fact a minimal form of DRO with overfitting control by cross-validation.

The Bonsai algorithm (Algorithm 2) first performs a predefined number of *discovery episodes*, using RERM to repeatedly solve an analogue of problem (4) that constructs

Algorithm 1 Robust Empirical Risk Minimization (RERM)

1: **Required:** datasets $D^k = \{(x_i^k, y_i^k)\}_{i=1}^{n^k}$, for $k = 1, \ldots, N$; model f; learning rate α
2: Randomly initialize f
3: **while** no overfit **do** `// By validation`
4: Train on datasets $D^1, \ldots, D^N$ by DRO: $f \leftarrow f - \alpha \cdot \nabla_f \left[\max_k \left(\frac{1}{|D^k|} \sum_{(x_i^k, y_i^k) \in D^k} \ell(f(x_i^k), y_i^k) \right)_{k=1}^{N} \right]$
5: **return** f

Algorithm 2 Bonsai algorithm (RFC)

Input: dataset D; the number of discovering rounds K
`// Discovery episodes`
$f_1 \leftarrow$ RERM($\{D\}$)
Split D into groups A_1, B_1 according to f_1. ($A_1 =$ examples correctly classified by f_1, $B_1 = D \backslash A_1$)
Available groups $P = \{A_1, B_1\}$
for $k \in [2, \ldots, K]$ **do**
  $f_k \leftarrow$ RERM(P)
  Split D into groups A_k, B_k according to f_k
  $P \leftarrow P \cup \{A_k, B_k\}$
`// Synthesis episode`
Pick a feature extractor function Φ, and K linear classifiers $\omega_1, ...\omega_k$ at random
Create K groups of pseudo-labels y^k by applying each f_k on D
$A = A_i \cap ... \cap A_K$
Update Φ, ω such that each pseudo-label y^k is well learned by the corresponding classifier ω_k and Φ: $\sum_{k=1}^{K} \frac{1}{|A|} \sum_{(x_i, y_i^k) \in A} \ell(\omega_k \circ \Phi(x_i), y_i^k) + \frac{1}{|D \backslash A|} \sum_{(x_i, y_i^k) \notin A} \ell(\omega_k \circ \Phi(x_i), y_i^k)$
return $\Phi, \{\omega_k\}_{k=1}^{K}$

a model f_k at each iteration, using an ensemble of subsets formed by selecting which examples were correctly or incorrectly recognized by the models $f_0 \ldots f_{k-1}$ constructed during the previous iterations.

The Bonsai algorithm performs a distillation-based *synthesis episode*. The goal is to learn a representation network $\Phi(x)$ such that we can emulate the functions f_k using a simple network with weights w_k on top of $\Phi(x)$. To that effect, we use the f_k models to compute pseudo-labels $y^k(x)$ for each example x. We then train a composite model with parameters Φ, w_1, ..., w_K whose k outputs are trained to replicate the pseudo-labels.

Why use linear classifiers in synthesis episode (line 11)?

The goal is to perform the synthesis step by distillation into a network whose architecture is as close as possible as the architecture of the "source" networks. However the distillation network needs one head for each source network. The least intrusive way to implement multiple heads is to duplicate the very last layer, hence linear. The opposite approach would be to claim the whole network is a classifier and the feature extractor is the identity. In this case, we can get a perfect synthesis loss (Alg 2, line 14) with an identity feature extractor which is obviously useless. We leave the non-linear classifier in *synthesis phase* as a future work.

What if the first RERM round achieves zero errors (line 3)?

The training set of the first RERM round is the union of the data associated with all OOD training environments. Since RERM avoids overfitting using a validation set (Alg 1, line 3), a perfect accuracy on both the merged training and validation sets means that the features discovered in the first round are already invariant in all training environments and perfectly predictive (100% accuracy). Therefore no further rounds are necessary since we already have a solution. This is in fact a degeneracy of the invariant training concept.

5. Experiments

This section presents experimental results that illustrate how the rich representations constructed with RFC can help the OoD performance and reduce the performance variance of OoD methods. Subsection 5.1 extends the experiments of Section 3 with RFC constructed representations. Subsection 5.2 compares RFC-initialized OoD methods with recent methods that aim for the second easiest-to-find representation. Subsection 5.3 reports results obtained on the CAMELYON17 dataset (Bandi et al., 2018) that is part of the WILDS benchmark suite (Koh et al., 2021). The final subsection provides additional empirical observation that casts light on the hyper-parameter tuning process and on the feature construction process itself.

5.1. Bonsai initialization helps all methods

All experiments reported in this section use the COLORMNIST task (Arjovsky et al., 2020) which consists of predicting labels that indicate whether the class of a colored digit image is less than 5 or not. The target label is noisy and only matches the digit class with probability 0.75 (correlation coefficient 0.5). Two training sets are provided where a spurious feature, the color of the digit, correlates with the target label with respective probabilities 0.8 and 0.9 (correlation coefficients 0.6 and 0.8). However, in the OoD testing set, the digit color is negatively correlated with the label (correlation coefficient -0.8). This testing protocol hits algorithms that rely on the spurious color feature because it happens to be more predictive than the robust feature in both training environments.

We compare six OoD training methods (IRMv1, vREx, SD, IGA, Fishr, CLOvE) and ERM after four types of initial-

	Rand	ERM	Bonsai	Bonsai-cf
IRMv1	54.0±2.4	68.9±1.1	66.5±1.5	**69.9±0.6**
vREx	53.1±2.0	69.3±0.7	**70.3±0.4**	69.9±0.4
SD	49.8±0.6	65.5±1.1	69.8±0.6	**70.4±0.4**
IGA	51.8±2.1	50.7±4.2	69.4±0.7	**70.0±0.8**
fishr	53.9±1.5	69.2±0.9	**70.2±0.4**	69.4±0.8
CLOvE	43.9±4.2	63.7±2.5	67.1±3.8	**68.4±0.8**
ERM	27.3±0.4	27.3±0.4	43.4±2.8	35.6±1.2
oracle	72.1 ± 0.2			

Table 1. OoD testing accuracy achieved on the COLORMNIST. The first six rows of the table show the results achieved by six OoD methods using respectively random initialization (Rand), ERM initialization (ERM), Bonsai initialization (Bonsai). The last column, (Bonsai-cf), reports the performance achieved by running the OoD algorithm on top of the frozen Bonsai representations. The seventh row reports the results achieved using ERM under the same conditions. The last row reminds us of the oracle performance achieved by a network using data from which the spurious feature (color) has been removed.

ization: (a) a random initialization with the popular Xavier method (Glorot & Bengio, 2010), (b) random initialization followed by several epochs of ERM, and (c) initialization with Bonsai representations, and (d) initialization with Bonsai representations that are subsequently frozen: the training algorithm is not allowed to update them (Bonsai-cf). The ERM initialization essentially consists of switching off the penalty terms defined by the various OoD method. This is comparable to the delicate penalty annealing procedures that are used by most authors (Arjovsky et al., 2020; Krueger et al., 2020; Pezeshki et al., 2020; Rame et al., 2021). The Bonsai initialization was computed by two discovery phase iterations.

For all six OoD algorithms and four initialization strategies, we select one of five penalization weights, $\{10, 50, 100, 500, 1000\}$ for the SD method, $\{1000, 5000, 10000, 50000, 100000\}$ for the other methods. For ERM initialization, we also select among five numbers of pretraining epochs $\{50, 100, 150, 200, 250\}$. These hyper-parameters were selected by peeking at the test set performance.[3] All experiments use the same 2-hidden-layers MLP network architecture (390 hidden neurons), Adam optimizer, learning rate=0.0005, L_2 weights regularization=0.0011 and binary cross-entropy objective function as the COLOREDMNIST benchmark (Arjovsky et al., 2020). Further details are provided in Appendix D.

[3]The small size of the COLOREDMNIST makes this hard to avoid. Tuning the hyper-parameters using the testing set favors in fact the ERM initialization because the test performance depends strongly on the number of pre-training epochs (Figure 1).

Methods	COLOREDMNIST	INVERSE COLOREDMNIST
IRMv1	69.9±0.6	80.3±2.2
vREx	69.9±0.4	84.0±1.2
SD	70.4±0.4	81.9±1.3
IGA	70.0±0.8	78.5±3.6
fishr	69.4±0.8	82.6±1.4
CLOvE	68.4±0.8	71.9±0.7
ERM	35.6±1.2	71.7±0.7
PI	70.9±0.3	51.0±4.7

Table 2. OoD test accuracy of PI and OoD/ERM methods on COLOREDMNIST and INVERSECOLOREDMNIST. The OoD/ERM methods use a frozen Bonsai representation (Bonsai-cf).

Table 1 reports the OoD testing accuracies obtained under these conditions. Bonsai initialization helps the OoD performance of most algorithms. Interestingly, the best results are achieved by freezing the Bonsai representation, which is consistent with the results of Section 3.2 showing that the OoD algorithm penalties are in fact insufficient to maintain the desired invariance constraints, even when initialized with the oracle weights. The bottom plot in Figure 2 shows that Bonsai initialization helps most OoD methods in this scenario as well, with the exception of the SD algorithm which penalizes the L_2 norm of logits in a manner that drives away the network from the oracle weights. Freezing the Bonsai representation doesn't prevent this from happening.

5.2. Aiming for the second easiest-to-find feature

Recent work (Liu et al., 2021; Nam et al., 2020; Bao et al., 2021; Ahmed et al., 2020; Creager et al., 2021) claims to achieve OoD generalization by discovering and using only the second easiest-to-find features. Although this strategy often works on datasets that were constructed to showcase OoD problems, the assumption that the second easiest features are the robust ones is unreasonable.

To illustrate this claim we construct a variant of the COLOREDMNIST dataset by changing the noise levels to make the robust feature (the digit shapes) more predictive than the spurious features (the digit color).

Table 2 compares the six OoD methods using the frozen Bonsai representation on both COLOREDMNIST and INVERSECOLOREDMNIST. All six methods achieve very comparable OoD testing accuracies. The ERM method fails on COLOREDMNIST but performs quite well on INVERSECOLOREDMNIST because relying on the most predictive features is a good strategy for this task. In contrast, the algorithm PI (Bao et al., 2021), which aims for the second easiest features, performs well on COLOREDMNIST but far worse on the easier INVERSECOLOREDMNIST task.

5.3. Bonsai initialization on a real-world task

The CAMELYON17 dataset (Bandi et al., 2018) contains histopathological images accompanied by a label indicating whether the central region of the image contains a tumor. The images were collected from five different hospitals with potentially different imaging hardware and different procedures. The WILDS benchmark (Koh et al., 2021) contains a task that uses this dataset with a very clear specification of which three hospitals are to be used as training data (302,436 images), which hospital is to be used for OoD generalization testing (85,054 images). The task also specifies multiple runs with different seeds in order to observe the result variability. Finally the task defines two ways to perform hyper-parameter selection: "IID Tune" selects hyper-parameters based on model performance on 33,560 images held out from the training data, "OoD Tune" selects hyper-parameters on the model performance observed on the fifth hospital (34,904 images).

We compare four different training methods, ERM, IRMv1, vREx, and CLOvE, using both ERM and Bonsai initialization with either two rounds (2-Bonsai) or three rounds (3-Bonsai) during the discovery phase. We also compare the effect of letting the training method tune the representation or freezing the representations obtained by the initialization procedure (ERM-cf, 2-Bonsai-cf, and 3-Bonsai-cf).

We strictly follow these procedures as well as the experimental settings suggested in the WILDS task. The network is a DenseNet121 model (Huang et al., 2017) trained by optimizing a cross-entropy loss with L_2 weight decay=0.01 using SGD with learning rate=0.001, momentum=0.9 and batch size=32. The penalty weights are selected from $\{0.5, 1, 5, 10, 50, 100, 500, 1000\}$ for IRMv1 and vREx, $\{0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1\}$ for CLOvE. The number of ERM pre-training iterations is selected in set from $\{0, 500, 1000, 5000, 10000\}$. Further details are provided in Appendix E.

Table 3 reports the OoD testing accuracies obtained by using both the IID and OoD hyper-parameter tuning approach. Accuracies were averaged over five repetitions with different random seeds. The first block of rows reports accuracies obtained by all four methods using ERM initialization. These accuracies come with large error bars because they considerably vary across repetitions. As a consequence, the accuracies differences observed in this block are not significant. The second block of rows shows that freezing the representations does not significantly improve this situation. In contrast, using a Bonsai representation with two discovery rounds (2-Bonsai) consistently improves the accuracies obtained by all four methods using either the IID or OoD tuning approaches (third block of rows). Freezing the Bonsai representation provides an additional boost (fourth block of rows).

Network Initialization	Methods	Test Acc IID Tune	Test Acc OoD Tune
$\times$	ERM	66.6±9.8	70.2±8.7
ERM	IRMv1	68.6±6.8	68.5±6.2
ERM	vREx	69.1±8.1	69.1±13.2
ERM	CLOvE	71.7±10.2	69.0±12.1
ERM-cf	ERM	$\times$	$\times$
ERM-cf	IRMv1	69.6±10.5	70.7±10.0
ERM-cf	vREx	69.6±10.5	70.6±10.0
ERM-cf	CLOvE	69.6±10.5	69.2±9.5
2-Bonsai	ERM	72.8±3.2	74.7±4.3
2-Bonsai	IRMv1	71.6±4.2	75.3±4.8
2-Bonsai	vREx	73.4±3.3	76.4±5.3
2-Bonsai	CLOvE	74.0±4.6	76.6±5.3
2-Bonsai-cf	ERM	78.2±2.6	78.6±2.6
2-Bonsai-cf	IRMv1	78.0±2.1	79.1±2.1
2-Bonsai-cf	vREx	77.9±2.7	79.5±2.7
2-Bonsai-cf	CLOvE	77.8±2.2	78.6±2.6
3-Bonsai-cf	ERM	72.9±5.3	73.3±5.3
3-Bonsai-cf	IRMv1	72.7±5.5	75.5±3.8
3-Bonsai-cf	vREx	72.7±5.4	75.1±5.3
3-Bonsai-cf	CLOvE	72.8±5.4	73.2±7.1

Table 3. Test Accuracy on the CAMELYON17 dataset. The hyper-parameter tuning process is performed on either the iid validation or the OoD validation set ("IID/OoD Tune"). We test ERM pre-trained initialization, 2-rounds, and 3-rounds Bonsai representation. As to the learning methods, we test ERM, IRMv1, vREx, and CLOvE. When freezing the representation and training the top-layer classifier only, we get "-cf" methods. The standard deviation is calculated on 5 random seeds [0-4].

Rosenfeld et al. (2022) claimed ERM may already discover enough features in the representation for OoD generalization. The second block in Table 3 shows the ERM learned representation is not rich enough in the CAMELYON17 case.

Using a Bonsai representation with three discovery rounds (3-Bonsai-cf) does not work as well. In fact, the features extracted during the third discovery phase round are not as predictive as the first two rounds (Table 4). More discovery rounds also increase the difficulty of the synthesis phase, as we want to distillate more features (including poor ones) into the same fixed-size representation.

Much to our surprise, Bonsai initialization consistently boosts the accuracies of both the ERM and OoD methods, using either the IID or OoD tuning method. The frozen Bonsai representations can even help ERM outperform earlier comparable results reported on the WILDS leaderboard[4] by about 5%.

[4] https://wilds.stanford.edu/leaderboard

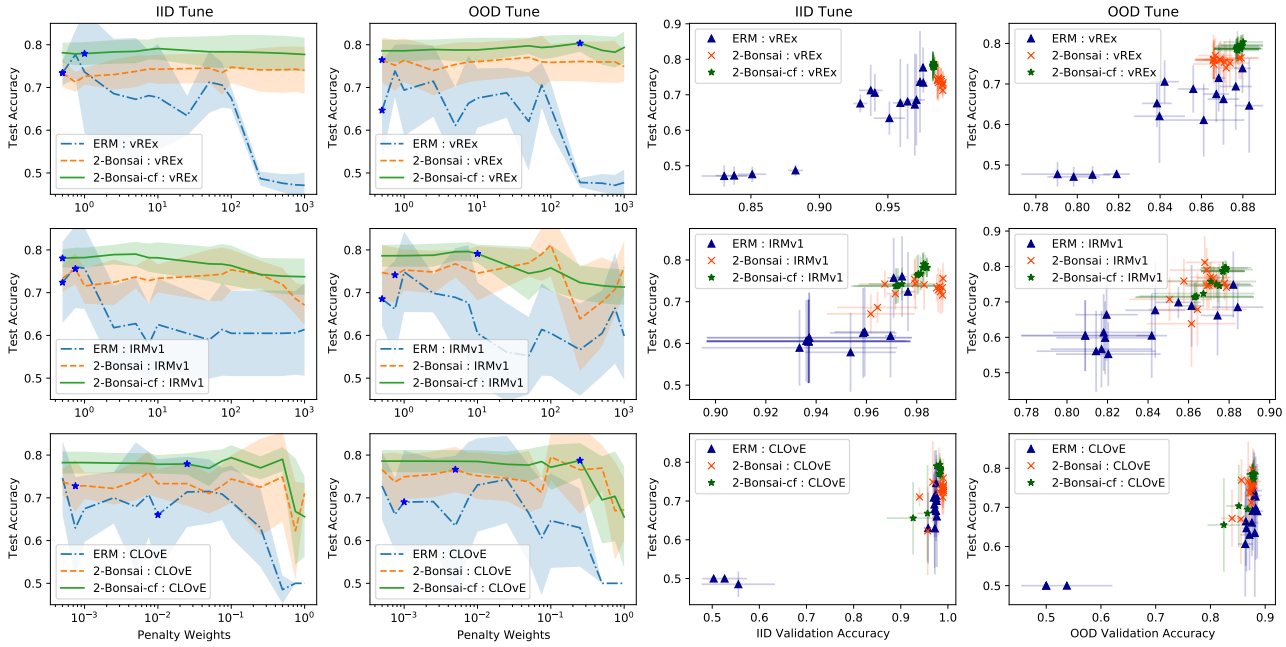

Figure 3. **Left half**: OoD testing accuracy as a function of the penalty weight. The six plots correspond to the IRMv1, vREx, and CLOvE algorithms with all other hyper-parameters selected using either the IID or OoD tuning method. Bonsai initialization makes these curves far more predictable than ERM initialization. Starts indicate the final penalty weight choice. **Right half**: OoD testing accuracy as a function of the validation accuracy. Bonsai initialization reduces the variance of both the IID and OoD validation performances, making them far more reliable indicators of the actual OoD testing performance.

5.4. Further observations

5.4.1. Hyper-parameter tuning

Figure 1 and 2 illustrate how the OoD generalization performance of many OoD methods depends strongly on hyper-parameters such as the number of pretraining epochs, the penalty weights, the learning epochs. This is in fact a consequence of the optimization-generalization dilemma itself. It is simply difficult to simultaneously ensure good OoD generalization performance and run a stable and efficient optimization process.

The left half of Figure 3 shows OoD testing accuracies for the CamelyOn17 task as a function of the penalty weights, with all other hyper-parameters chosen using either the IID or OoD tuning method. With ERM pretraining, the OoD testing performance of all three OoD methods (IRMv1, vREx, CLOvE) depends very chaotically on the penalty weight. In contrast, with a frozen Bonsai representation, the OoD testing performance of OoD methods, as a function of the penalty weight, follows a much smoother curve.

The right half of Figure 3 shows the relation between the IID/OoD validation accuracies and the OoD testing accuracies for three OoD methods using both ERM and Bonsai initialization. Bonsai initialization reduces the variance of both the IID and OoD validation performances, making them far more reliable indicators of the actual OoD testing performance.

5.4.2. The value of the synthesis episode

The ColoredMNIST and InverseColoredMNIST experiments (Table 2) show that the robust feature can be discovered during different rounds of the discovery phase. We can therefore wonder whether the discovery phase already produces the correct invariant representation during one of its successive rounds.

This is not the case in general. Table 4 reports the OoD testing accuracies of the classifiers constructed during the first three rounds of the discovery phase. All three accuracies are substantially worse than the accuracies achieved by any algorithm using a frozen 2-Bonsai-cf representation (Table 3). This indicates that these higher accuracies are obtained by simultaneously exploiting features discovered by different rounds of the discovery phase. Making them all simultaneously available is indeed the role of the synthesis phase.

6. Conclusion

This work makes several contributions:

- We point out the severity of the optimization-

Round 1	Round 2	Round 3
66.6±9.8	73.2±5.7	61.8±10.2

Table 4. OoD test accuracies for the models constructed by the first three discovery rounds for the CAMELYON17 task. The first round amounts to performing ERM. The second round extracts a useful set of features. The third round extracts comparatively weaker features. All these accuracies remain substantially worse than those achieved by training a system on top of the combined representation computed during the synthesis phase (see Table 3).

generalization dilemma in the OoD setup, showing that the various penalties introduced by OoD methods are either too strong to optimize or too weak to achieve their goals.

- We propose to work around the problem by seeding the networks with a rich representation that contains a diversity of features readily exploitable by the algorithm. We formalize this objective, and we describe an algorithm, the Bonsai algorithm, that constructs such rich representations.
- We show that Bonsai initialization helps a variety of OoD methods achieve a better OoD testing performance. Interestingly, when we additionally prevent the learning algorithm from modifying the Bonsai representations, we not only observe a further boost in the performance of OoD methods, but we also raise the performance of ERM to the same level, substantially outperforming previous comparable results on the CAMELYON17 dataset for examples.
- Finally, we also show that Bonsai initialization facilitates both IID and OoD variants of hyper-parameter tuning and model selection.

Therefore, it appears that the inductive bias that comes with a broad set of diverse features brings considerable benefits to the various invariant/OoD training methods proposed in the recent literature.

7. Acknowledgements

The authors acknowledge support from the National Science Foundation (NSF Award 1922658) and from the Canadian Institute for Advanced Research (CIFAR).

References

Ahmed, F., Bengio, Y., van Seijen, H., and Courville, A. Systematic generalisation with group invariant predictions. In *International Conference on Learning Representations*, 2020.

Arjovsky, M., Bottou, L., Gulrajani, I., and Lopez-Paz, D. Invariant risk minimization. *arXiv*, 2020.

Bae, J.-H., Choi, I., and Lee, M. Meta-learned invariant risk minimization. *arXiv*, 2021.

Bandi, P., Geessink, O., Manson, Q., Van Dijk, M., Balkenhol, M., Hermsen, M., Bejnordi, B. E., Lee, B., Paeng, K., Zhong, A., et al. From detection of individual metastases to classification of lymph node status at the patient level: the camelyon17 challenge. *IEEE Transactions on Medical Imaging*, 2018.

Bao, Y., Chang, S., and Barzilay, R. Predict then interpolate: A simple algorithm to learn stable classifiers, 2021.

Beery, S., Van Horn, G., and Perona, P. Recognition in terra incognita. In *ECCV*, 2018.

Ben-Tal, A., Ghaoui, L. E., and Nemirovski, A. *Robust Optimization*, volume 28 of *Princeton Series in Applied Mathematics*. Princeton University Press, 2009.

Creager, E., Jacobsen, J.-H., and Zemel, R. Environment inference for invariant learning. In *International Conference on Machine Learning*, pp. 2189–2200. PMLR, 2021.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In Teh, Y. W. and Titterington, M. (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

Gu, X., Feng, J., Sun, J., and Xu, Z. Domain-free adversarial splitting for domain generalization, 2021.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Huang, Z., Wang, H., Xing, E. P., and Huang, D. Self-challenging improves cross-domain generalization. *arXiv*, 2020.

Ilyas, A., Santurkar, S., Tsipras, D., Engstrom, L., Tran, B., and Madry, A. Adversarial examples are not bugs, they are features. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman,

K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Kamath, P., Tangella, A., Sutherland, D. J., and Srebro, N. Does invariant risk minimization capture invariance? *AISTATS*, 2021.

Koh, P. W., Sagawa, S., Marklund, H., Xie, S. M., Zhang, M., Balsubramani, A., Hu, W., Yasunaga, M., Phillips, R. L., Gao, I., Lee, T., David, E., Stavness, I., Guo, W., Earnshaw, B. A., Haque, I. S., Beery, S., Leskovec, J., Kundaje, A., Pierson, E., Levine, S., Finn, C., and Liang, P. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.

Koyama, M. and Yamaguchi, S. Out-of-distribution generalization with maximal invariant predictor. *arXiv*, 2020.

Krueger, D., Caballero, E., Jacobsen, J.-H., Zhang, A., Binas, J., Priol, R. L., and Courville, A. Out-of-distribution generalization via risk extrapolation (rex). *arXiv*, 2020.

Kumar, A., Sarawagi, S., and Jain, U. Trainable calibration measures for neural networks from kernel mean embeddings. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2805–2814. PMLR, 10–15 Jul 2018.

Liu, E. Z., Haghgoo, B., Chen, A. S., Raghunathan, A., Koh, P. W., Sagawa, S., Liang, P., and Finn, C. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pp. 6781–6792. PMLR, 2021.

Nam, J., Cha, H., Ahn, S., Lee, J., and Shin, J. Learning from failure: Training debiased classifier from biased classifier. *arXiv*, 2020.

Nichol, A., Achiam, J., and Schulman, J. On first-order meta-learning algorithms. *arXiv*, 2018.

Pezeshki, M., Kaba, S. O., Bengio, Y., Courville, A., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. *arXiv*, 2020.

Rame, A., Dancette, C., and Cord, M. Fishr: Invariant gradient variances for out-of-distribution generalization. *arXiv preprint arXiv:2109.02934*, 2021.

Rosenfeld, E., Ravikumar, P., and Risteski, A. Domain-adjusted regression or: Erm may already learn features sufficient for out-of-distribution generalization. *arXiv preprint arXiv:2202.06856*, 2022.

Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *ICLR*, 2019.

Sagun, L., Evci, U., Uğur Güney, V., Dauphin, Y., and Bottou, L. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv*, 2017.

Shi, Y., Seely, J., Torr, P. H., Siddharth, N., Hannun, A., Usunier, N., and Synnaeve, G. Gradient matching for domain generalization. *arXiv preprint arXiv:2104.09937*, 2021.

Simons, S. Minimax theorems and their proofs. In Du, D.-Z. and Pardalos, P. M. (eds.), *Minimax and Applications*, pp. 1–23, Boston, MA, 1995. Springer US.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

Teney, D., Abbasnejad, E., Lucey, S., and van den Hengel, A. Evading the Simplicity Bias: Training a Diverse Set of Models Discovers Solutions with Superior OOD Generalization. *arXiv*, 2021.

Wald, Y., Feder, A., Greenfeld, D., and Shalit, U. On calibration and out-of-domain generalization. *arXiv preprint arXiv:2102.10395*, 2021.

A. MAML-IRM resembles vREx+Fish

We omit the MAML-IRM method in our experiments because we can show that minimizing its cost amounts to minimizing a mixture of the vREx and Fish costs.

Notations:

- $\mathcal{E}$: indicates a set of environments.
- θ: indicates the model parameters.
- $L_i(\theta)$: indicates an ERM loss (e.g. MSE, cross-entropy) of a model parameterized by θ on environments i.
- $\bar{g_i} = L_i^{'}(\theta)$: is the gradients of $L_i(\theta)$.
- $\bar{H_i} = L_i^{''}(\theta)$: is the Hessian of $L_i(\theta)$.

Let $U_i(\theta) = \theta - \alpha L_i^{'}(\theta)$ denote the updated parameters after performing a SGD iteration on environments i. The MAML-IRM loss can be expressed as:

$$L_{\text{maml-irm}} = \mathbb{E}_s[L_j(U_i(\theta))] + \lambda\sqrt{Var_s[L_j(U_i(\theta))]} \tag{5}$$

where the notation $\mathbb{E}_s$ and Var_s respectively denote the average and the variance with respect to all pairs of distinct environment $s = \{i, j | i \in \mathcal{E}, j \in \mathcal{E}, i \neq j\}$, and where λ is a hyper-parameter.

According to (Nichol et al., 2018), the gradients of the first term is:

$$\frac{\partial(\mathbb{E}_s[L_j(U_i(\theta))])}{\partial\theta} = \mathbb{E}_s[\bar{g_j} - 2\alpha\bar{H}_i\bar{g_j}] + O(\alpha^2) \tag{6}$$

Note that $\mathbb{E}_s[-2\bar{H}_i\bar{g_j}] = \mathbb{E}_s[-\frac{\partial\langle g_i, g_j\rangle}{\partial\theta}]$ is in fact the gradients of $-\langle g_i, g_j\rangle$, the Fish penalty.

We now turn out attention to the second term $\sqrt{Var_s[L_j(U_i(\theta))]}$. Expanding $L_j(U_i(\theta))$ with a Taylor series gives:

$$\begin{aligned}
L_j(U_i(\theta)) &= L_j(\theta) + \left\langle L_j^{'}(\theta), (U_i(\theta) - \theta)) \right\rangle + O(\alpha^2) && (7)\\
&= L_j(\theta) - \alpha \left\langle L_i^{'}(\theta), L_j^{'}(\theta) \right\rangle + O(\alpha^2) \qquad (\longleftarrow U_i(\theta) = \theta - \alpha L_i^{'}(\theta)) && (8)\\
&= L_j(\theta) - \alpha \langle \bar{g_i}, \bar{g_j} \rangle + O(\alpha^2) && (9)
\end{aligned}$$

Therefore

$$\begin{aligned}
Var_s(L_j(U_i(\theta))) &= Var_s[L_j(\theta) - \alpha \langle \bar{g_i}, \bar{g_j} \rangle] + O(\alpha^2) && (10)\\
&= Var_s[L_j(\theta)] + \alpha^2 Var_s[\langle \bar{g_i}, \bar{g_j} \rangle] - 2\alpha \text{Cov}_s[L_j(\theta), \langle \bar{g_i}, \bar{g_j} \rangle] + O(\alpha^2)\\
&= Var_s[L_j(\theta)] - 2\alpha \text{Cov}_s[L_j(\theta), \langle \bar{g_i}, \bar{g_j} \rangle] + O(\alpha^2)\\
&= Var_s[L_j(\theta)] - 2\alpha\{\mathbb{E}_s[L_j(\theta) \langle \bar{g_i}, \bar{g_j} \rangle] - \mathbb{E}_s[L_j(\theta)]\mathbb{E}_s[\langle \bar{g_i}, \bar{g_j} \rangle]\} + O(\alpha^2)\\
&= Var_s[L_j(\theta)] - 2\alpha\mathbb{E}_s[(\frac{L_i(\theta) + L_j(\theta)}{2} - \mathbb{E}[L(\theta)]) \langle \bar{g_i}, \bar{g_j} \rangle] + O(\alpha^2)
\end{aligned}$$

The first term of this expression, $Var_s[L_j(\theta)]$, penalizes a high variance of the loss across environments. It is equal to the vREx penalty. The second term, $-2\alpha\mathbb{E}_s[(\frac{L_i(\theta)+L_j(\theta)}{2} - \mathbb{E}[L(\theta)]) \langle \bar{g_i}, \bar{g_j} \rangle]$ is a weighted average of $\langle g_i, g_j \rangle$, that is a smoothed Fish penalty.

In conclusion, optimizing the MAML-IRM cost amounts to optimizing a λ controlled mixture of the vREx and Fish costs.

B. GroupDRO interpolates environments while vREx extrapolates.

The vREx objective function can be expressed as:

$$L_{\text{vrex}} = \mathbb{E}_{e\in\mathcal{E}}(L_e) + \lambda Var_{e\in\mathcal{E}}(L_e) \tag{11}$$

The GroupDRO objective function is a mixture of the per-environment costs L_e with positive coefficients:

$$L_{\text{groupDRO}} = \mathbb{E}_{e\in\mathcal{E}}(p_e L_e) \tag{12}$$

where the adjustable mixture coefficients $p_e \geq 0$, $\sum_{e\in\mathcal{E}} p_e = 1$, are treated as constaants for computing the gradients $\frac{\partial L_e}{\theta}$.

The gradient of these two cost functions are:

$$\frac{\partial L_{\text{vrex}}}{\theta} = \mathbb{E}_{e\in\mathcal{E}}([2\lambda(L_e - \mathbb{E}_i L_i) + 1]g_e) \tag{13}$$

$$\frac{\partial L_{\text{groupDRO}}}{\theta} = \mathbb{E}_{e\in\mathcal{E}}(p_e g_e) \tag{14}$$

where $g_e = \frac{\partial L_e}{\theta}$ is the gradients of network weights θ on environment e.

Because the p_e mixture coefficients are always positive, it is easy to see that GroupDRO follows a direction aligned with a convex combination of the per-environment gradients. In contrast, vREx can follow a direction that is outside this convex hull because the coefficients $\mathbb{E}_i L_i) + 1]$ can be positive or negative).

C. Loss landscape of OoD methods

Here we visualize the loss landscape of some of OoD penalties on a synthetic two-dimensional problem, TwoBits, which was introduced by (Kamath et al., 2021) as a simplified version of the COLOREDMNIST. TwoBits is a binary classification problem $Y = \pm 1$ with two binary inputs $X_1 = \pm 1$ and $X_2 = \pm 1$ distributed as follows:

$$\begin{aligned} Y &\sim \text{Rademacher}(0.5) \\ X_1 &\sim Y \cdot \text{Rademacher}(\alpha_e) \\ X_2 &\sim Y \cdot \text{Rademacher}(\beta_e) \end{aligned}$$

where Rademacher(α) denotes the law of a random variable taking value -1 with probability α and taking $+1$ probability $1-\alpha$. The training algorithms observe two training environments, $(\alpha_e, \beta_e) \in \{(0.1, 0.1), (0.1, 0.3)\}$. The four input patterns (X_1, X_2) are represented by four points $\{\Psi(1,1), \Psi(-1,-1), \Psi(1,-1), \Psi(-1,1)\}$ in the representation space where Ψ can represent any network architectures with numerical outputs. Following (Kamath et al., 2021), we use a mean squared loss and focus on the symmetric case $\Psi(-x) = -\Psi(x)$. The representation space can therefore be displayed with only two dimensions, $\Psi(1,-1) = -\Psi(-1,1)$ and $\Psi(1,1) = -\Psi(-1,-1)$.

Figure 4 shows a heat map of the penalty terms of three OoD methods (IRMv1, vREx, SD) as a function of the chosen representation. The stars denote three solutions: (a) the Invariant solution which only uses feature X_1 because this is the feature whose correlation with the label remains the same across the training environments, (b) the ERM solution which uses both features, and (c) a random feature initialization with small variance for which the representations $\Psi(1,1), \Psi(1,-1)$ are close to zero.

All three OoD methods have low penalties when the $\Psi(1,1), \Psi(1,-1)$ are close to zero. This explains why random initialization performs so poorly with these methods. In contrast, pretraining with ERM leads to a new initialization point that is away from the origin and close to the ERM solution. The OoD performance then depends on the existence of a good optimization path between this initialization and the Invariant solution. Alas Figure 4 shows a lot of optimization difficulties such as finding a solution that lies at the bottom of an elongated ravine (ill-conditioning). In conclusion, the impact of the number of ERM pretraining epochs is essentially unpredictable.

D. Experimental details for the ColoredMNIST experiments

We use the original COLOREDMNIST dataset (Arjovsky et al., 2020) with two training environments $(0.25, 0.1), (0.25, 0.2)$. The target label correlates with the invariant feature (the digit shape) with a probability 0.75. The sirious feature (color) correlates with the target label with a probability 0.8 and 0.9, respectively. Each training environment contains 25000 images where the size of each image is $2 \times 14 \times 14$. For all COLOREDMNIST experiments, we use a fully connected neural network with 3 layers (392 (input dim) $\times 390 \times 390 \times 1$), trained with the Adam optimizer with learning rate 0.0005. We use a L2 weights regularization with parameter 0.0001 for INVERSECOLOREDMNIST tasks and 0.0001 in the regular

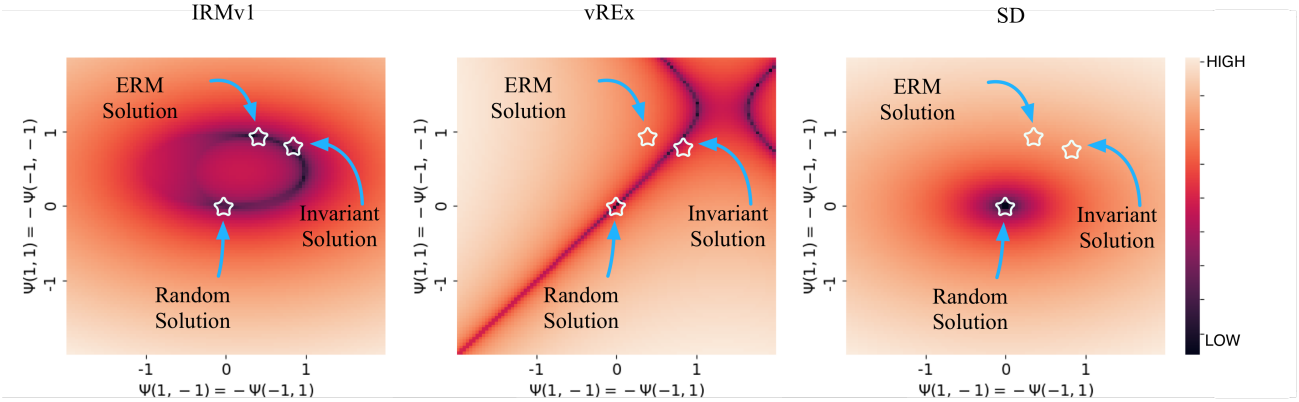

Figure 4. The IRMv1, vREx, and SD landscapes show a challenging non-convex landscape in the two-dimensional TwoBits problem. In particular, the path between the ERM solution and the invariant solution often involves climbing the loss landscape.

COLOREDMNIST tasks. For the CLOvE method, we use a Laplacian kernel $k(r, r_0) = exp(\frac{-|r-r_0|}{0.4})$ (Kumar et al., 2018) with mini-batch size 512. All other methods train using full batches. For the ERM baseline and for computing the oracle performance, we search the L2 regularization parameter in $\{0.0001, 0.0005, 0.001, 0.005, 0.01\}$. We run each experiment 10 times to get the standard deviation.

D.1. Hyper-parameter searching space

Table D.1 shows the penalty weights searching space for all OoD methods in the COLOREDMNIST experiments. Table 6 shows the training epochs searching space for different OoD methods and network initialization/representation on the COLOREDMNIST dataset.

Table 5. Penalty weight search space for both the COLOREDMNIST and INVERSECOLOREDMNIST datasets.

	COLOREDMNIST	INVERSECOLOREDMNIST
IRMv1	$10000 \times \{0.1, 0.5, 1, 5, 10\}$	$10000 \times \{0.1, 0.5, 1, 5, 10\}$
vREx	$10000 \times \{0.1, 0.5, 1, 5, 10\}$	$10000 \times \{0.1, 0.5, 1, 5, 10\}$
IGA	$10000 \times \{0.1, 0.5, 1, 5, 10\}$	$10000 \times \{0.1, 0.5, 1, 5, 10\}$
CLOvE	$10000 \times \{0.1, 0.5, 1, 5, 10\}$	$10 \times \{0.1, 0.5, 1, 5, 10\}$
Fishr	$10000 \times \{0.1, 0.5, 1, 5, 10\}$	$10000 \times \{0.1, 0.5, 1, 5, 10\}$
SD	$100 \times \{0.1, 0.5, 1, 5, 10\}$	$\{0.05, 0.1, 0.5, 1, 5\}$
RSC	$(0.995, 0.98) \times \{0.95, 0.97, 0.98, 0.99, 1\}$	-
LfF	$\{0.1, 0.2, 0.3, 0.4, 0.5\}$	-
Fish	$0.001 \times \{0.1, 0.5, 1, 5, 10\}$	-

D.2. Bonsai algorithm

For all COLOREDMNIST experiments, we use a 2-rounds Bonsai *discovery phase* trained with respectively 50 and 500 epochs. Then we train 500 epochs for the distillation network of the Bonsai *synthesis phase*. For the INVERSECOLOREDMNIST experiments, we again use a 2-rounds Bonsai *discovery phase* trained with respectively 150 and 400 epochs. We choose these training epochs because they can maximize the IID validation performance during each round.

D.3. PI training

We use the original implementation from PI (Bao et al., 2021). Because the PI algorithm is closely related to the *discovery phase*, we use the same hyper-parameters and settings.

Table 6. The number of training epochs search space for the COLOREDMNIST dataset, with $i \in [0, 24]$.

	Rand/ERM	Bonsai	Bonsai-cf
IRMv1	$i \times 20$	$i \times 2$	$i \times 125$
vREx	$i \times 20$	$i \times 2$	$i \times 20$
IGA	$i \times 20$	$i \times 1$	$i \times 20$
CLOvE	$i \times 30$	$i \times 1$	$i \times 20$
Fishr	$i \times 20$	$i \times 1$	$i \times 20$
SD	$i \times 20$	$i \times 1$	$i \times 20$
RSC	$i \times 1$	-	-
LfF	$i \times 20$	-	-
Fish	$i \times 20$	-	-

E. Experimental details for the CAMELYON17 experiments

We strictly follow the implementation of the CAMELYON17 task in the WILDS benchmark (Koh et al., 2021). For the results presented in section 5.4.1, we additionally search the penalty weights in the set $\{0.5, 0.75, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000\}$ for IRMv1 and vREx methods, and the set $\{0.5, 0.75, 1, 2.5, 5, 7.5, 10, 25, 50, 75, 100, 250, 500, 750, 1000\} \times 10^{-3}$. The CLOvE method require a kernel function, we choose the Laplacian kernel $k(r, r_0) = exp(\frac{-|r-r_0|}{l})$ (Kumar et al., 2018) where l is a positive scalar. For the CLOvE baseline with an ERM pretrained initialization (the fourth row of table 3), we test the scalar $l \in \{0.1, 0.2\}$ and choose the better one $l = 0.2$. For the other CLOvE experiments on CAMELYON17, we choose $l = 0.1$.

We train the *synthesis phase* 20 epochs and the other methods/phase 10 epochs. Hyper-parameter tuning strictly follows the IID and OoD tuning process described in the WILDS task. We use a L2 weights regularization $1e-6$ during the *synthesis phase* to help it get a lower training loss on the pseudo-labels. During any further training that updates the weights of the learned representation, we keep the L2 weights regularization to be the same as $1e-6$. Otherwise, a stronger L2 weights regularization will destroy the learned representation. We also tried other L2 regularization weights in $\{1e-2, 1e-4, 1e-6\}$. Table 8 shows the synthesis quality with different (*synthesis phase*) L2 weights decay. Two smaller L2 weights decay hyper-parameters $\{1e-4, 1e-6\}$ can arrive at a good synthesis quality. The corresponding test performances on the frozen representation "2-Bonsai-cf" of the two smaller hyper-parameters are higher too (Table 7). Table 7 shows that the "2-Bonsai-cf" representation can also reliability gain a high performance once the synthesis quality is good.

After the *synthesis phase*, RFC provides us a rich representation Φ and K linear classifiers $\omega_1, \ldots, \omega_K$. In the downstream tasks, such as OoD/ERM training, we will keep the representation Φ and initialize the top-layer classifier ω. There are at least two ways to initialize it: 1) initialize ω as the average of $\omega_1, \ldots, \omega_K$ with the hope that the initial top-layer classifier uses all discovered features. 2) randomly initialize ω. Table 9 shows the test performance of OoD/ERM methods with each top-layer initialization method. None of the two top-layer initialization methods significantly outperforms the other one. We choose the first top-layer initialization method in all main experiments because of the interpretation.

Table 7. Test accuracy of OoD methods (IRMv1, vREx) and ERM methods. Three *synthesis phase* L2 weights decay $\{1e{-}2, 1e{-}4, 1e{-}6\}$ are tested. All the other settings are the same as the main results in Table 3.

Synthesis phase L2 weights decay	Network Initialization	Methods	Test Acc	
			IID Tune	OoD Tune
$1e-6$	2-Bonsai-cf	ERM	78.2±2.6	78.6±2.6
$1e-6$	2-Bonsai-cf	IRMv1	78.0±2.1	79.1±2.1
$1e-6$	2-Bonsai-cf	vREx	77.9±2.7	79.5±2.7
$1e-4$	2-Bonsai-cf	ERM	77.8±1.7	78.8±2.3
$1e-4$	2-Bonsai-cf	IRMv1	77.7±1.7	78.9±2.3
$1e-4$	2-Bonsai-cf	vREx	77.9±1.7	79.7±1.7
$1e-2$	2-Bonsai-cf	ERM	75.2±7.8	75.5±7.4
$1e-2$	2-Bonsai-cf	IRMv1	75.0±7.9	75.4±7.5
$1e-2$	2-Bonsai-cf	vREx	75.4±7.7	75.8±7.3

Table 8. The train and IID-validation performance of the *synthesis phase*. Note that it uses the pseudo-labels instead of the true labels as Y. Three *synthesis phase* L2 weights decay $\{1e-2, 1e-4, 1e-6\}$ are tested.

(Synthesis phase) L2 weights decay	Train accuracy	IID-validation accuracy
$1e-6$	99.7±0.0	97.4±0.3
$1e-4$	99.6±0.1	97.4±0.2
$1e-2$	93.9±0.7	94.9±0.5

Table 9. Test performance of IRMv1, vREx, and ERM methods on a 2 rounds Bonsai representation. The top-layer classifier is initialized by either the average of $\omega_1, \ldots \omega_K$ (Average) or a random initialization (Random). When freezing the representation and training the top-layer classifier only, we get the "-cf" methods.

Network Initialization	Methods	Average		Random	
		IID Tune	OOD Tune	IID Tune	OOD Tune
2-Bonsai	ERM	72.8±3.2	74.7±4.3	73.0±3.7	75.9±6.7
2-Bonsai	IRMv1	71.6±4.2	75.3±4.8	74.5±2.3	75.2±6.5
2-Bonsai	vREx	73.4±3.3	76.4±5.3	73.0±3.9	77.1±5.0
2-Bonsai-cf	ERM	78.2±2.6	78.6±2.6	77.8±2.4	78.6±2.6
2-Bonsai-cf	IRMv1	78.0±2.1	79.1±2.1	78.0±2.1	79.1±2.1
2-Bonsai-cf	vREx	77.9±2.7	79.5±2.7	78.0±2.6	79.7±2.4