Learning explanations that are hard to vary

Giambattista Parascandolo[1, 2,*] **Alexander Neitz**[1,*]

Antonio Orvieto[2] **Luigi Gresele**[1, 3] **Bernhard Schölkopf**[1, 2]

[1] MPI for Intelligent Systems, Tübingen
[2] ETH, Zürich
[3] MPI for Biological Cybernetics, Tübingen

Abstract

In this paper, we investigate the principle that *good explanations are hard to vary* in the context of deep learning. We show that averaging gradients across examples – akin to a logical OR ($\vee$) of patterns – can favor memorization and 'patchwork' solutions that sew together different strategies, instead of identifying invariances. To inspect this, we first formalize a notion of consistency for minima of the loss surface, which measures to what extent a minimum appears only when examples are pooled. We then propose and experimentally validate a simple alternative algorithm based on a logical AND ($\wedge$), that focuses on invariances and prevents memorization in a set of real-world tasks. Finally, using a synthetic dataset with a clear distinction between invariant and spurious mechanisms, we dissect learning signals and compare this approach to well-established regularizers.

1 Introduction

Consider the top of Figure 1, which shows a view from above of the loss surface obtained as we vary a two dimensional parameter vector $\theta = (\theta_1, \theta_2)$, for a fictional dataset containing two observations x_A and x_B. Note the two global minima at coordinates $(\theta_1, \theta_2) = (-0.5, -0.5)$ and $(0.8, 0.8)$. Depending on the initial values of θ — marked as white circles — gradient descent converges to one of the two minima. Judging solely by the value of the loss function, which is zero in both cases, the two minima look equally "good".

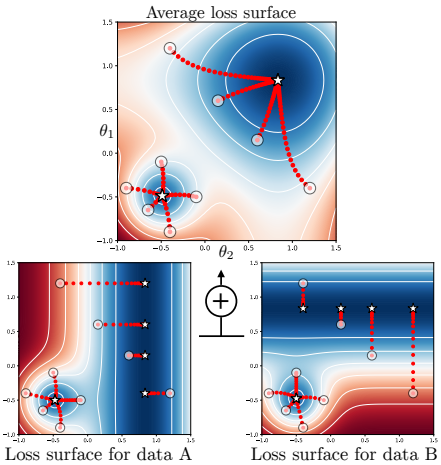


Figure 1: Loss landscapes of a two-parameter model. Averaging gradients forgoes information that can identify patterns shared across different environments.

However, looking at the loss surfaces for x_A and x_B separately, as shown below, a crucial difference between those two minima appears: Starting from the same initial parameter configurations and following the gradient of the loss, $\nabla_\theta \mathcal{L}(\theta, x_i)$, the probability of finding the same $(0.8, 0.8)$-minimum in either case is zero. In contrast, the minimum in the lower-left corner has a significant overlap across the two loss surfaces, so gradient descent can converge to it even if training on x_A (or x_B) only. Note that after averaging there is no way to tell what the two loss surfaces looked like: *Are we destroying information that is potentially important?*

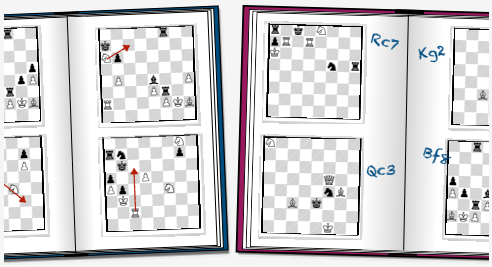
*Equal contribution. Correspondence to `gparascandolo@tue.mpg.de` and `aneitz@tue.mpg.de`

Preprint. Under review.

In this paper, we argue that the answer is yes. In particular, we hypothesize that if the goal is to find *invariant mechanisms* in the data, these can be identified by finding explanations (e.g. model parameters) that are hard to vary across examples.

The notion of invariance requires something that remains the same, as something else changes. We will consider a setting where we assume that the data comes from different *environments*: An invariant mechanism is shared across all environments and generalizes out of distribution (o.o.d.), but it might be hard to model; each environment also has spurious explanations that are easy to spot (i.e. 'shortcuts'), but do not generalize o.o.d.

An intuitive toy example Imagine two second-hand books of chess puzzles. As shown on the right, we can learn to solve them based on the two independent shortcuts (red arrows for the left book *OR* hand-written solutions on the right), or actually learn to play chess (the invariant mechanism). While both strategies work on other problems from the same books (i.i.d.), only the latter will generalise to new chess puzzles books (o.o.d.). How to distinguish the two? We characterize this as follows: We would have not learned about the red arrows had we trained on the book on the right, and vice versa with the hand-written notes.



We formalize a notion of *consistency*, which characterizes to what extent a minimum of the loss surface appears *only* when data from different environments are pooled. Minima with low consistency are 'patchwork' solutions, which (we hypothesize) sew together different strategies and should not be expected to generalize to new environments. An intuitive description of this principle was proposed by physicist David Deutsch: *"good explanations are hard to vary"* [Deutsch, 2011].

Using the notion of consistency, we define *Invariant Learning Consistency* (ILC), a measure of the expected consistency of the solution found by a learning algorithm on a given hypothesis class. The ILC can be improved by changing the hypothesis class or the learning algorithm, and in the last part of the paper we focus on the latter. We then analyse why current practices in deep learning provide little incentive for networks to learn invariances, and show that standard training is instead set up with the explicit objective of greedily maximizing speed of learning, i.e., progress on the training loss. When learning "as fast as possible" is not the main objective, we show we can trade-off some "learning speed" for prioritizing learning the invariances. A practical instantiation of ILC leads to o.o.d. generalization on a challenging synthetic task where several established regularizers fail to generalize; moreover, following the memorization task from Zhang et al. [2016], ILC prevents convergence on CIFAR-10 with random labels, as no shared mechanism is present, and similarly when a portion of training labels is incorrect. Lastly, we set up a behavioural cloning task based on the game CoinRun [Cobbe et al., 2018], and observe better generalization on new unseen levels.

2 Explanations that are hard to vary

We consider a collection of datasets $\{\mathcal{D}^e\}_{e\in\mathcal{E}}$ with $|\mathcal{E}| = d$ and $\mathcal{D}^e = (x_i^e, y_i^e)$, $i_e = 1, \dots, n_e$. Here $x_i^e \in \mathcal{X} \subseteq \mathbb{R}^m$ is the vector containing the observed patterns, and $y_i^e \in \mathcal{Y} \subseteq \mathbb{R}^p$ the targets. The superscript $e \in \mathcal{E}$ indexes some aspect of the data collection process, and can be interpreted as an environment. Our objective is to infer a function $f : \mathcal{X} \to \mathcal{Y}$ connecting x_i^e to y_i^e, which corresponds to a *mechanism* shared across all environments (see introduction). Concretely, f may be parametrized by a neural network with continuous activations, and therefore lives in a finite-dimensional subspace of the vector space of continuous functions: $f \in \mathcal{F} \subseteq C^0(\mathbb{R}^m, \mathbb{R}^p)$. For network weights $\theta \in \Theta \subseteq \mathbb{R}^n$, we denote the neural network output at $x \in \mathcal{X}$ as $f_\theta(x)$.

Gradient-based optimization. To find an appropriate model f_θ, standard optimizers, such as Gradient Descent [Cauchy, 1847] or Adam [Kingma and Ba, 2014], rely on gradients from a *pooled* loss function $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$. This function measures the *average* performance of the neural network when predicting data labels, measured across all environments:

$$\mathcal{L}(\theta) := \frac{1}{|\mathcal{E}|} \sum_{e\in\mathcal{E}} \mathcal{L}_e(\theta), \quad \text{with} \quad \mathcal{L}_e(\theta) := \frac{1}{|\mathcal{D}^e|} \sum_{(x_i^e, y_i^e)\in\mathcal{D}^e} \ell(f(x_i^e; \theta), y_i^e); \tag{1}$$

where $\ell : \mathbb{R}^p \times \mathbb{R}^p \to [0, +\infty)$ is usually chosen to be the $L2$ loss or the cross-entropy loss. In this setting, vanilla gradient descent (GD) reads $\theta^{k+1}_{\text{GD}} = \theta^k_{\text{GD}} - \eta \nabla \mathcal{L}(\theta^k_{\text{GD}})$, where $\eta > 0$ is the so-called learning rate. Under some standard technical assumptions [Lee et al., 2016], $(\theta^k_{\text{GD}})_{k \geq 0}$ converges to a parameter $\text{GD}_\infty(\theta^0) := \lim_{k \to \infty} \theta^k_{\text{GD}}$, which is a local minimizer of $\mathcal{L}$, with probability one[2].

When do we *not* learn invariances? Towards our goal of finding invariances, we start by describing factors that might prevent this in practice.

(i) Training stops once the loss is low enough. If by the time the network converged it learned spurious patterns, invariances will not be learned anymore. This depends on the *rate* at which different patterns are learned. The rates at which invariant patterns emerge (and vice-versa, the spurious patterns do not) can be improved by: (a) careful architecture design, e.g. as done by hardcoding spatial equivariance in convolutional networks; (b) fine-tuning models pre-trained on large amounts of data, where strong features already emerged and can be readily selected.

(ii) Learning signals: everything looks relevant for a dataset of size 1. Backpropagation efficiently computes gradients for every example *independently*. Informally, the signal computed from each example is identical to the one for an equivalent dataset of size 1, where therefore every pattern looks potentially relevant to the task. To find invariant patterns *across* examples, if we compute our training signals on each of them *independently*, we have to rely on the way these are aggregated.[3]

(iii) Aggregating gradients: averaging maximizes learning speed. The default method to pool gradients is to take their *arithmetic mean*. GD applied to $\mathcal{L}$ is designed to minimize the pooled loss *as fast as possible*[4]. Indeed, a step of GD is equivalent to finding a tight[5] quadratic upper bound $\hat{\mathcal{L}}$ to $\mathcal{L}$, and then jumping to the minimizer of this approximation [Nocedal and Wright, 2006]. While speed is often desirable, note that by construction GD completely ignores one potentially crucial piece of information: The gradient $\nabla \mathcal{L}$ is the result of averaging signals $\nabla \mathcal{L}_e$, which correspond to the patterns visible from the single environments at this stage of the optimization procedure. In other words, gradient descent with average gradients *greedily maximizes for learning speed*, but in some situations we would *trade some convergence speed for invariance*. For instance, instead of performing an arithmetic mean between gradients (logical OR), we might want to look towards a logical AND, which can be characterized as a *geometric* mean.[6]

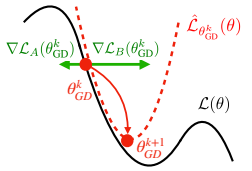


Figure 2: GD step can be independent of the inconsistency in gradient directions.

In Sec. 2.3 we elaborate on this idea and on the possible implementations of a logical AND between gradients. Before presenting this discussion, we take some time to better motivate the need for invariant learning consistency and to construct a precise and non-ambiguous definition of consistency.

2.1 Formal definition of ILC

Consider an iterative algorithm $\mathcal{A}$ with access to environments $\mathcal{E}$. Starting from $\theta^0 \in \Theta \subseteq \mathbb{R}^n$ with distribution $p(\theta^0)$, we can write $\theta^{k+1}_{\mathcal{A}} = \mathcal{A}((\theta^i_{\mathcal{A}})_{0 \leq i \leq k}, \mathcal{E})$. Assuming, for simplicity of exposition, that $\mathcal{A}$ is deterministic, we define its *convergence point*[7] $\mathcal{A}_\infty(\theta^0) := \lim_k \theta^k_{\mathcal{A}}$. We denote by $\mathcal{A}_\infty(\theta^0, \mathcal{J})$ the outcome of algorithm $\mathcal{A}$ when trained on the environments $\mathcal{J} \subseteq \mathcal{E}$.

[2]Similar results hold for SGD [Daneshmand et al., 2018, Kushner and Yin, 2003], momentum [Jin et al., 2017], and adaptive methods including Adam [Barakat and Bianchi, 2019] and Adagrad [Ward et al., 2019].

[3]After computing the gradients for a dataset of $n-1$ examples, if an n-th example appeared, we would just compute one more vector of gradients and add it to the sum. A Gaussian Process for example would require recomputing the entire solution from scratch, as all interactions are considered.

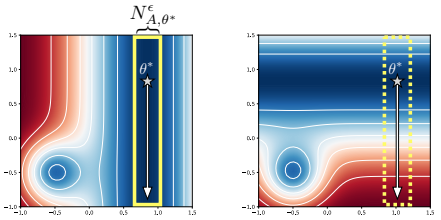
[4]The same reasoning holds for SGD in the finite-sum optimization case $\mathcal{L} = \frac{1}{m} \sum_{i=1}^m \mathcal{L}_i$, where gradients from a mini-batch are seen as unbiased estimators of gradients from the pooled loss. [Bottou et al., 2018].

[5]Assume that $\mathcal{L}$ has L-Lipshitz gradients (i.e. curvature bounded form above by L). Then, at any point $\tilde{\theta}$, we can construct the upper bound $\hat{\mathcal{L}}_{\tilde{\theta}}(\theta) = \mathcal{L}(\tilde{\theta}) + \nabla \mathcal{L}(\tilde{\theta})^\top (\theta - \tilde{\theta}) + L \|\theta - \tilde{\theta}\|^2 / 2$.

[6]Fig. 1 shows how a sum can be seen as a logical OR: the two orthogonal gradients from data A and data B at (0.5,0.5) point to different directions, yet both are kept in the combined gradient: $(0, 1) + (1, 0) = (1, 1)$. Loosely speaking, a sum is large if any of the summands is large, a product is large if all of the factors are large.

[7]Existence of this limit can be guaranteed for standard (stochastic) optimization algorithms using Doob's martingale convergence theorem (see e.g. Prop. 2.4. in Wang et al. [2017]).

Consistency of a convergence point. Let $\Theta^*_{\mathcal{A}}$ be the set of convergence points of $\mathcal{A}$ when trained on *all* available experiments (pooled data). That is, $\Theta^*_{\mathcal{A}} = \{\theta^* \in \Theta \mid \exists\ \theta^0 \in \mathbb{R}^n \text{ s.t. } \mathcal{A}_\infty(\theta^0, \mathcal{E}) = \theta^*\}$. For instance, if $\mathcal{A}$ is gradient descent, the result of Lee et al. [2016] implies that $\Theta^*_{\mathcal{A}}$ is the set of local minimizers of the pooled loss $\mathcal{L}$. To each $\theta^* \in \Theta^*_{\mathcal{A}}$, we want to associate a consistency score, quantifying the concept "*good θ^* are hard to vary*". In other words, we would like the score to capture the consistency of the loss landscape around θ^* across the different environments. For example, in Fig. 1 the loss landscape near the bottom-left minimizer is consistent across environments, while the top-right minimizer is not. In order to derive a formal definition to capture the phenomenon we just described, let us consider the parameters in θ^*, output by $\mathcal{A}$ optimizing the pooled loss: we want to characterize the landscape around this point from the perspective of a fixed environment $e \in \mathcal{E}$. To do this, we define the set N^ϵ_{e,θ^*} to be the largest path-connected region of space containing both θ^* and the set $\{\theta \in \Theta \,|\mathcal{L}_e(\theta) \leq \mathcal{L}_e(\theta^*) + \epsilon\ \}$, where $\epsilon > 0$ is a fixed small positive parameter. In other words, we require that if $\theta \in N^\epsilon_{e,\theta^*}$ then there exist a continuous path from θ^* to θ in parameter space where each parameter also is in N^ϵ_{e,θ^*}. The set N^ϵ_{e,θ^*} will thus contain those parameters θ around θ^* such that the loss of environment e in θ is at least comparable to the loss of environment e in θ^*. From the perspective of environment e, all these points are equivalent to θ^*. Once this set is constructed, we would like to evaluate its elements with respect to a different environment $e' \neq e$. We will say that $e' \in \mathcal{E}$ is consistent with $e \in \mathcal{E}$ in θ^* if $\max_{\theta \in N^\epsilon_{e,\theta^*}} \left|\mathcal{L}_{e'}(\theta) - \mathcal{L}_e(\theta)\right|$ is small. Repeating this reasoning for all environment pairs, we arrive at the following *consistency score*.



Loss surface for data A Loss surface for data B

$$\mathcal{C}^\epsilon(\theta^*) := -\max_{(e,e')\in\mathcal{E}^2} \max_{\theta \in N^\epsilon_{e,\theta^*}} |\mathcal{L}_{e'}(\theta) - \mathcal{L}_e(\theta)|. \tag{2}$$

This consistency is our formalization of the principle "*good explanations are hard to vary*". Finally, we can write down an invariant learning consistency score for $\mathcal{A}$:

$$\text{ILC}(\mathcal{A}, p_{\theta^0}) := \mathbb{E}_{\theta^0 \sim p(\theta^0)} \left[\mathcal{C}^\epsilon(\mathcal{A}_\infty(\theta^0))\right]. \tag{3}$$

That is, the learning consistency of an algorithm measures the expected consistency across environments of the minimizer it converges to.

Approximating the consistency score. A cheap yet effective approximation of $\mathcal{C}^\epsilon(\theta^*)$ can be computed using a second-order Taylor expansion of $\mathcal{L}_e$ around θ^*. Indeed, subject to this approximation, $\theta \in N^\epsilon_{e,\theta^*}$ becomes a *quadratic constraint*. This implies we can approximate $\mathcal{C}^\epsilon(\theta^*)$ by solving a sequence of maximization problems with simple convex[8] constrains, solvable with accelerated mirror descent [Krichene et al., 2015]. A cruder yet faster approximation can be constructed by substituting the Taylor expansions directly into the objective $|\mathcal{L}_{e'}(\theta) - \mathcal{L}_e(\theta)|$.

A classic example of a patchwork solution. One well known version of the universal approximation function theorem [Cybenko, 1989] shows that a one-hidden-layer neural network with sigmoid activations and enough neurons can approximate any function $f^* : [0, 1] \to \mathbb{R}$. In the appendix we show how the constructive proof used to obtain the weights (i.e. tiling the input space with groups of neurons), leads to a maximally inconsistent solution according to $\mathcal{C}^\epsilon(\theta^*)$, which indeed one would not expect to generalize o.o.d.

2.2 ILC is linked to the "logical AND" between landscapes

In this section we draw a direct connection between our theoretical characterization of consistency and the motivation around the "logical AND" of patterns presented at the beginning of Sec. 2. For the sake of clarity, here we consider only two environments (A and B) and assume $\theta^* \in \Theta^*_{\mathcal{A}}$ to be a local minimizer (with loss zero) of the pooled loss $\mathcal{L}$ as well as of the single environment losses $\mathcal{L}_A$ and $\mathcal{L}_B$ (like in Fig. 1).

[8]This is assuming we have positive definite Hessian at θ^* for each experiment. This is true in the context of Figure 1, but was also shown to hold for the overparametrized neural network case in late training: see e.g. Fig. 3 in [Yao et al., 2019] and Fig. 1 in [Sagun et al., 2017]. However, for the sake of robustness, we suggest to regularize the numerically computed Hessians by adding a small positive diagonal matrix.

Without loss of generality, let us assume $\theta^* = 0 \in \mathbb{R}^n$; by Taylor expansion, we get an approximation[9] for small $\|\theta\|$: $\mathcal{L}(\theta) \approx \frac{1}{2}\theta^\top H_{A+B}\theta$, where $H_{A+B} = \frac{1}{2}(H_A + H_B)$ is the *arithmetic mean* of the Hessians (matrix of second derivatives computed at θ^*) H_A and H_B, which approximate the shape of the local minimizer θ^* when training in different environments (the level sets are ellipses with equation $\theta^\top H\theta = c$). H_{A+B} in Fig. 3 is constructed without looking at the single geometric properties of landscape A or B: it performs a "logical OR" between the landscape dominant eigendirections. In contrast, the Karcher mean $H_{A\wedge B}$ [Ando et al., 2004] models the inconsistency between the geometry of A and B, yielding an ellipse with 100 times smaller volume — it finds the shared geometry. In the appendix we show how the consistency of θ^* can be linked to the difference between geometric and arithmetic mean. More precisely, we prove that $\mathcal{C}^\epsilon(\theta^*) \geq -2\epsilon\left(\frac{\det(H_{A+B})}{\det(H_{A\wedge B})}\right)^2$. Since, as for the geometric mean of positive numbers, $0 \leq \det(H_{A\wedge B}) \leq \det(H_{A+B})$; we conclude that the consistency is largest when the *shapes* of A and B are the same — this is exactly what happens at the consistent bottom-left minimizer of Fig. 1.

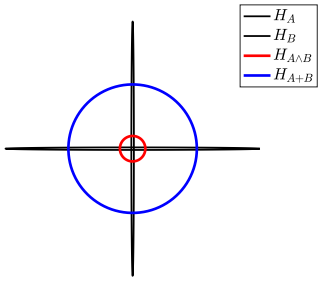


Figure 3: $H_A = \text{diag}(0.01, 1)$ and $H_B = \text{diag}(1, 0.01)$. The geometric average retains the volume of the original ellipses, while the volume of H_{A+B} is 100 times bigger. This magnification indicates that landscape A is not consistent with landscape B.

2.3 From local to global: masking gradients with a logical AND

Here we seek an algorithm $\mathcal{A}$ which has a low *ILC-regularized loss*, i.e.

$$\mathcal{L}^{\text{ILC}}(\mathcal{A}, p_{\theta^0}) = \mathbb{E}_{\theta^0\sim p(\theta^0)}\left[\frac{1}{|\mathcal{E}|}\sum_{e\in\mathcal{E}}\mathcal{L}_e(\mathcal{A}_\infty(\theta^0))\right] - \lambda\cdot\text{ILC}(\mathcal{A}, p_{\theta^0}), \tag{4}$$

where $\lambda > 0$ reflects how much we care about finding invariant patterns in our problem. For $\lambda = 0$, we recover standard gradient descent. As consistency is a *local* property of the minimum, we need to somehow extend this information to the learning process when we are still far from the minima. Our strategy will be essentially the following: *To converge to a consistent minimum, we only take "consistent" steps, corresponding to the "logical AND" of patterns coming from different experiments*, as also motivated by the characterization of ILC in Sec. 2.2: In the approximate local setting of Fig. 3, gradient descent on the pooled loss $\frac{1}{2}\theta^\top H_{A+B}\theta$ is much faster than gradient descent on the geometrically averaged loss $\frac{1}{2}\theta^\top H_{A\wedge B}\theta$. Hence, following the gradient from the logical AND of landscapes should reduce the attraction power of inconsistent minimizers, increasing the ILC.

The AND-mask. We translate the qualitative reasoning we just presented to an algorithmic insight for increasing the consistency of gradient-based optimizers. We will refer to this as the *AND-mask*. Intuitively, in its most simple implementation[10], we zero out those gradient components with respect to weights that have *inconsistent signs* across environments. Formally, the masked gradients at iteration k are $m_t(\theta^k)\odot\nabla\mathcal{L}(\theta^k)$, where $m_t(\theta^k)$ vanishes for any component where there are less than $t \in \{d/2+1, \dots, d\}$ agreeing gradient signs across environments (d is the number of environments in the batch), and is equal to one otherwise. For convenience, our implementation of the AND-mask uses a *threshold* $\tau \in [0, 1]$ as hyper-parameter instead of t, such that $t = \frac{d}{2}(\tau + 1)$. Mathematically, for every component $[m_\tau]_j$ of m_τ, $[m_\tau]_j = \mathbb{1}\left[\tau d \leq |\sum_e \text{sign}([\nabla\mathcal{L}_e]_j)|\right]$, where $\mathbb{1}$ is the Boolean indicator function.

As a first result, we show that following the AND-masked gradient leads to convergence in the directions made visible by the AND mask. The proof is presented in the appendix and is a simple adaptation of the analysis of non-convex smooth gradient descent [Ghadimi and Lan, 2013].

[9] Even though this approximation might not hold when θ gets far away from θ^*, it provides us a useful simplified perspective. Indeed, this *quadratic model* is heavily used in the optimization community (see e.g. Jastrzębski et al. [2017], Zhang et al. [2019a], Mandt et al. [2017]) and is linked to important quantities in information geometry and statistics [Amari, 2016], such as the Fisher information matrix.

[10] We believe many other implementations (for instance leveraging on quantities like a signal to noise ratio) of our reasoning are possible. Here, for the sake of brevity, we focus on a very simple one.

Proposition 1. Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ have L-Lipschitz gradients and consider AND-masked GD with learning rate $\eta \leq 1/L$. After k iterations, AND-masked GD visits at least once a point θ where $\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(1/k)$.

Note that the AND-mask has the same time and space complexity of standard non-convex gradient descent, i.e., linear in the number of examples that we average. Due to its simplicity and computational efficiency, this is the instantiation of ILC that we will use in the experiment section.

Asymptotic behaviour in the face of randomness. Here we put the AND mask through a theoretical test: For gradients coming from different environments that are inconsistent (or even random), how fast does the AND mask reduce the magnitude of the step taken in parameter space, compared to standard gradient descent? *In case of inconsistency, the AND mask should quickly make the gradient steps more conservative.*

To make our derivation more transparent, while keeping the setting interesting, we consider a fixed set of n parameters θ and assume each $\nabla\mathcal{L}_e(\theta) \in \mathbb{R}^n$ is drawn independently from a multivariate Gaussian with zero mean and $\sigma^2 I$ covariance.

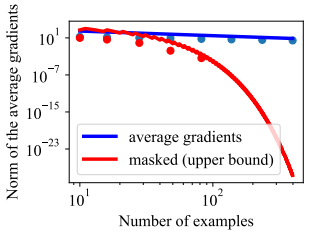


Figure 4: Magnitude of gradient step (average gradients or masked) ($\sigma = 1$, $|\theta| = 3000$, $t = 0.8d$).

Proposition 2. Consider the setting we just outlined, with $\mathcal{L} = (1/|\mathcal{E}|)\sum_{e\in\mathcal{E}} \mathcal{L}_e$ and $d := |\mathcal{E}|$. While $\mathbb{E}\|\nabla\mathcal{L}(\theta)\|^2 = \mathcal{O}(n/d)$, we have that for any $t \in \{d/2+1, \ldots, d\}$ there exists $1 < c \leq 2$ such that $\mathbb{E}\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(n/c^d)$.

The proof is presented in the appendix, and an illustration with numerical verification in Fig. 4 (the magnitudes of masked gradients (•) for more than 100 examples were always zero in the numerical verification). Intuitively, in the presence of purely random patterns, the AND-mask decreases the strength of these signals exponentially fast, as opposed to linearly.

3 Experiments

Real-world datasets are generated by (causal) generative processes which have been argued to share mechanisms [Pearl, 2009]. However, mechanisms and spurious signals are often entangled, making it hard to assess what part of the learning signal is due to either. As the goal of this paper is to dissect these two components to understand how they ultimately contribute to the learning process, we create a simple synthetic dataset that allows us to minutely control the complexity, intensity, and number of shortcuts present in the data. After that, to evaluate whether spurious signals can be detected even in high dimensional networks and datasets, we test the AND-mask on a memorization task similar to the one proposed in Zhang et al. [2016], and a similar task with label noise. Finally, we set up a behavioural cloning task on CoinRun, a game with procedurally generated levels. All details are in the appendix.

3.1 The synthetic memorization dataset

We introduce a dataset for a binary classification task. The input dimensionality is $d = d_M + d_S$. $p(y|x_{d_M})$ is the same across *all* environments (i.e. the *mechanism*). $p(y|x_{d_S})$ is *not the same* across all environments (the *shortcuts*). While the mechanism is shared, it needs a highly non-linear decision boundary to classify the data. The shortcuts are not shared across environments, but provide a simple way to classify the data, even when pooling all the environments together. See Figure 5 for a concrete example with d_M and d_S equal to 2, and two environments (A and B). The spirals (on d_M) are

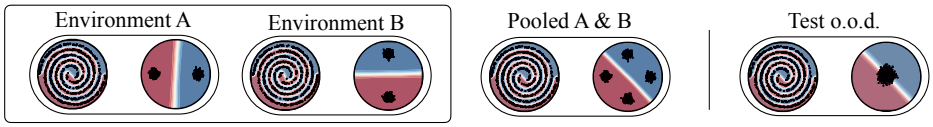


Figure 5: A 4-dimensional instantiation of the synthetic memorization dataset for visualization. Every example is a dot in both circles, and it can be classified by finding either of the "oracle" decision boundaries shown.

invariant but hard to model. The shortcuts (on d_S) are simple blobs but different in every environment: in A, linearly separable through a vertical decision boundary, in B with a horizontal one. If the two environments are pooled, a new decision boundary emerges as the most 'natural' one on the shortcut dimensions, i.e. the diagonal one. While this perfectly classifies data in both environments A and B, critically *it would have not been found by training on either partition A or B alone*. The o.o.d. test data, has the same mechanism but random shortcuts, therefore any method relying exclusively on the shortcuts for its decisions will have chance-level performance out of distribution. Finally, we set up a high-dimensional version of the synthetic dataset as described above. All details about the dataset, baselines, training curves, are reported in the appendix.

Despite the apparent simplicity of this dataset, note that it is challenging to find the invariant mechanism. In high dimensions, even with tens of pooled environments, the shortcuts allow for a *simple* classification rule under almost every classical definition of 'simple': the boundary is *linear*, it has a *large margin*, it can be expressed with *small weights*, it is *fast to learn*, robust to input noise, and has *perfect accuracy and no i.i.d. generalization gap*. Finding the complex decision boundary of the spirals, instead, is a fiddly process and arguably a much slower path towards small loss.

Baselines. We test several baselines, all multilayer perceptrons, which include some of the most common regularizers used in deep learning — Dropout, L1, L2, Batch normalization. We select hyperparameters randomly from Table 1 in the appendix. Note that these baselines are all agnostic to the division into different environments. We also consider more complex baselines that explicitly make use of the environment labels, namely: *(i)* Domain Adversarial Neural Networks (DANN) [Ganin et al., 2016], a method specifically designed to address domain adaptation by obfuscating the domain information with an adversarial classifier; and *(ii)* Invariant Risk Minimization (IRM) [Arjovsky et al., 2019], which we discuss in detail in the appendix. The AND-mask is trained with the same configurations used in Table 1.

Results. In Figure 6, we show training and test accuracy. Interestingly, for DANN even a single linear layer is sufficient to align the distributions from different environments using only the shortcuts, such that they become indistinguishable to the domain-discriminating classifier. The AND-mask was the only method to achieve perfect test accuracy, essentially by ignoring the shortcuts and modeling the spirals instead. In particular, the combination of the AND-mask with L1 or L2 regularization gave the most robust results overall, as they help suppress neurons that at initialization are naturally tuned towards the shortcuts.

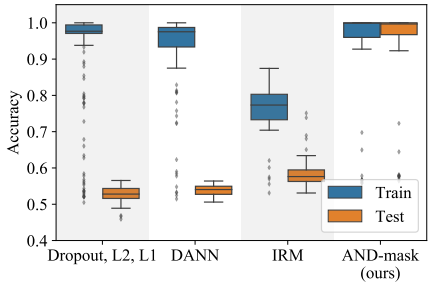


Figure 6: Results on the synthetic dataset.

Correlations between average, memorization and generalization gradients. Given the synthetic nature of the dataset, we can intervene on its data-generating process to take a closer look at the learning signal coming from the mechanisms and from the shortcuts. We can isolate the two, and measure via correlation their contribution to the average gradients for different random initializations. Figure 7 shows a larger correlation between the original average gradients and the memorization gradients (orange lines), than between the average gradients and the oracle generalization gradients (blue lines). While the signal from the mechanisms is present in the original average gradients, its magnitude is smaller and it is 'drowned' by the memorization signal. Instead, (right panel) the AND-mask effectively suppresses most memorization gradients due to the shortcuts.

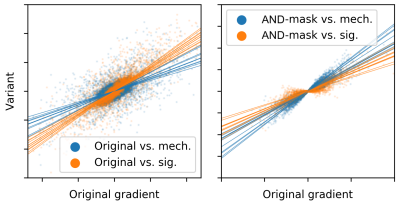


Figure 7: Gradient correlations.

3.2 Experiments on CIFAR-10

Memorization in a vision task. Zhang et al. [2016] showed that neural networks trained with standard regularizers — like L2 and Dropout — can still memorize large training datasets with *shuffled* labels, i.e. reaching ≈100% training accuracy. Their experiments raised significant questions

about the generalization properties of neural networks and the role of regularizers in constraining the hypothesis class. Our hypothesis is that ILC — for example implemented as the AND-mask — should prevent memorization on a similar task with the shuffled labels, as gradients will tend to largely 'disagree' in the absence of a shared mechanism. However, when the labels are *not shuffled*, ILC should have a much weaker effect, as real shared mechanisms are still present in the data.

To test our hypothesis, we ran an experiment that closely resembles the one in [Zhang et al., 2016] on CIFAR-10. We trained a ResNet[11] on CIFAR-10 with *random labels*, with and without the AND-mask. In all experiments we used batch size 80, and treated each example as its own "environment". Recall that standard gradient averaging is equivalent to an AND-mask with threshold 0. As shown in Figure 8, the ResNet with standard average gradients memorized the data, while slightly increasing the threshold for the AND-mask quickly prevented memorization (dark blue line). In contrast, training the same networks on the dataset *with the original labels* resulted in both of them converging and generalizing to the test set, confirming that the mask did not significantly affect the generalization error with a general underlying mechanism in the data.

Finally, note that there is no standard notion of environments in CIFAR-10, which is why we treated every example as coming from its own environment. This assumption is not unreasonable, as every image in the dataset was literally collected in a different physical environment. If anything, it is the standard i.i.d. assumption that hides this variety behind a notion of a single distribution encompassing all environments. The results of this experiment further support this interpretation, and can serve as evidence that — in some cases — we might be able to identify invariances even without an explicit partition into environments, as this can be already identified at the level of individual examples.

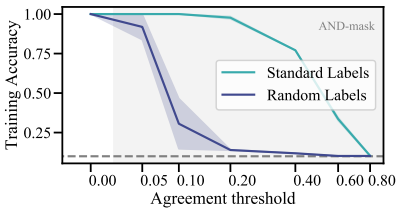


Figure 8: As the AND-mask threshold increases, memorization on CIFAR-10 with random labels is quickly hindered.

Label noise. Following up on the previous experiment, we further test how the AND-mask performs in the presence of label noise, i.e. when a portion of the labels in the training set are randomly shuffled (25% here). According to our hypothesis, gradients computed on examples with random labels should disagree and get masked out by the AND-mask, while signal from correctly labeled data should contribute to update the model. As shown in Figure 9, the performance on the *incorrectly* labeled portion of the dataset is well *below* chance for the AND-mask (as it predicts correctly despite the wrong labels), while the baseline again memorizes the incorrect labels. On the test set (with untouched labels), the baseline peaks early then decreases as the model overfits, while the AND-mask slowly but steadily improves.

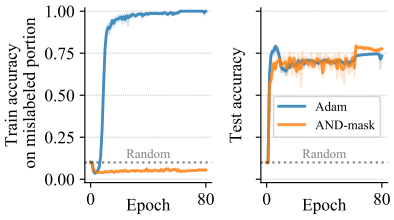


Figure 9: The AND-mask prevents overfitting to the incorrectly labeled portion of the training set (left) without hurting the test accuracy.

3.3 Behavioral Cloning on CoinRun

CoinRun [Cobbe et al., 2019b] is a game introduced to test how RL agents generalize to novel situations. The agent needs to collect coins in a 2D environment, while jumping on top of walls and boxes, and avoiding enemies.[12] Each level is procedurally generated — i.e. it has a different combination of sprites, background, and layout — but the physics and goals are invariant. The experiments presented in [Cobbe et al., 2019b] showed that state-of-the-art RL algorithms fail to model these invariant mechanisms, performing poorly on new levels unless they are trained on tens of thousands of them. To test our hypothesis, we set up a behavioral cloning task using CoinRun.[13]

[11] We used the fast ResNet architecture from Fomin et al. [2020], all details in the appendix.

[12] See Figure 17 in the appendix for a visualization of the game.

[13] In order to obtain a robust evaluation, here we preferred to approach behavioral cloning instead of the full RL problem, since it is a standard supervised learning task and has substantially fewer moving parts than most deep RL algorithms.

We start by pre-training a strong policy π^* using standard PPO [Schulman et al., 2017] for 400M steps on the full distribution of levels (see appendix for all details). We then generate a dataset of pairs $(s, \pi^*(a|s))$ from the on-policy distribution. The training data consists of 1000 states from each of 64 levels. The test data is collected on 2000 different levels. A ResNet-18 $\hat{\pi}_\theta$ is then trained to minimize the loss $D_{\mathrm{KL}}(\pi^*||\hat{\pi}_\theta)$ on the training set.

We compare the generalization performance of regular Adam to a version that uses the AND-mask. For each method we ran an automatic hyperparameter optimization study using Tree-structured Parzen Estimation [Bergstra et al., 2013] of 1024 trials. Despite the theoretical computational efficiency of computing the AND-mask as presented in Section 2.3 (i.e., linear time and memory in the size of the mini-batch, just like classic SGD), current deep learning frameworks like PyTorch [Paszke et al., 2017] have optimized routines that sum gradients across examples in a mini-batch before it is possible to efficiently compute the AND-mask. For this reason, here we use the AND-mask in a slightly different way. During training, in each iteration we sample a batch of data from a randomly chosen level out of the 64 available (and cycle through them all once per epoch). We then apply the AND-mask 'temporally', only allowing gradients that are consistent across time (and therefore across levels). See Algorithm 1 in the appendix for a detailed description of this alternative formulation of the AND-mask. The results in Figure 10 show the minimum test loss for the top-10 runs, supporting the hypothesis that the AND-mask helps identify invariant mechanisms across different levels.

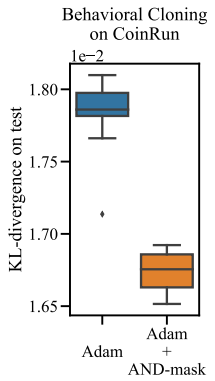


Figure 10

4 Related Work

The classic formalization of the problem of generalization [Vapnik, 1995] is concerned with the case of independent and i.i.d. samples. Covariate Shift [Sugiyama et al., 2007, Quionero-Candela et al., 2009, Sugiyama and Kawanabe, 2012] formalizes the case in which the distribution of the covariates at test time differs from that observed at training. Standard solutions to the problem involve re-weighting the training examples, but require the additional assumption that the supports of the two distributions are overlapping.

Following Pearl [2009], we assume the existence of a causal model giving rise to each observed distribution. An important feature of its causal factorization is that many of the conditionals, which we can think of as physical mechanisms underlying the statistical dependencies represented, are expected to remain *invariant* under interventions or changing external conditions. This postulate has appeared in various forms in the literature [Haavelmo, 1943, Simon, 1953, Hurwicz, 1962, Pearl, 2009, Schölkopf et al., 2012].[14] Based on this insight, it was proposed that regression based on causal features should presents desirable invariance and robustness properties [Mooij et al., 2009, Schölkopf et al., 2012, Peters et al., 2016, Rojas-Carulla et al., 2018, Heinze-Deml et al., 2018, von Kügelgen et al., 2018]. Thus learning the mechanisms may help achieve a stable performance across a number of conditions. Interesting ideas on learning invariances in the data have appeared in recent work Peters et al. [2016], Subbaswamy et al. [2018], Heinze-Deml and Meinshausen [2017], Arjovsky et al. [2019]. In the appendix we present a more detailed comparison to IRM [Arjovsky et al., 2019].

ILC can be used in a setting of domain generalization [Muandet et al., 2013], but it is not limited to it. As demonstrated in the experiments in Section 3.2, the AND-mask can be applied even where domain labels are not available (as it is the case for CIFAR-10). By treating every example as coming from its own domain, modern domain generalization methods relying on domain classifiers (like DANN Ganin et al. [2016] or Balaji et al. [2018]) would need as many output units as there are training examples (i.e. 50'000 for CIFAR-10).

Looking at agreement to characterize and improve the representations learned by neural networks is an idea that has been explored in recent work, mostly looking at the cosine similarity between gradients [Du et al., 2018, Eshratifar et al., 2018, Fort et al., 2019, Zhang et al., 2019b]. We did not look at the cosine similarity here, mainly for two reasons: *(i)* It is a 'global' property of the gradients, and as such it would not allow us to extract precise information about different patterns in the network; *(ii)* It is difficult to scale to more than two vectors, and the cost of computing pairwise interactions scales quadratically with the number of examples used.

[14] This would be different for a non-causal factorization of the joint distribution, see Schölkopf [2019]

5 Conclusions

Generalizing out of distribution is one of the most significant open challenges in machine learning, and relying on invariances across environments or examples may be key in certain contexts. In this paper we analyzed how neural networks trained by averaging gradients across examples might converge to solutions that ignore the invariances, especially if these are harder to learn than spurious patterns. We argued that if learning signals are collected *on one example at the time* — as it is the case for gradients, e.g., computed with backpropagation — the way these signals are aggregated can play a significant role in the patterns that will ultimately be expressed: Averaging gradients in particular can be too permissive, acting as a *logical OR* of a collection of distinct patterns, and leading to a 'patchwork' solution. We introduced and formalized the concept of Invariant Learning Consistency, and showed how to learn invariances even in the face of alternative explanations that — although spurious — fulfill most characteristics of a good solution. The AND-mask is but one of multiple possible ways to improve consistency, and it is unlikely to be a practical algorithm for all applications. However, we believe this should not distract from the general idea which we are trying to put forward — namely, that it is worthwhile to study learning of explanations that are *hard to vary*, with the longer term goal of advancing our understanding of learning, memorization and generalization.

Acknowledgments

The authors wish to thank Sebastian Gomez and Luca Biggio for proof-reading the paper, and Nando de Freitas for fruitful discussions in the early stage of this project. The authors thank the Max Planck ETH Center for Learning Systems for supporting Giambattista Parascandolo, and the International Max Planck Research School for Intelligent Systems for supporting Alexander Neitz.

References

S.-i. Amari. *Information geometry and its applications*, volume 194. Springer, 2016.

T. Ando, C.-K. Li, and R. Mathias. Geometric means. *Linear algebra and its applications*, 385: 305–334, 2004.

M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

Y. Balaji, S. Sankaranarayanan, and R. Chellappa. Metareg: Towards domain generalization using meta-regularization. In *Advances in Neural Information Processing Systems*, pages 998–1008, 2018.

A. Barakat and P. Bianchi. Convergence analysis of a momentum algorithm with adaptive step size for non convex optimization. *arXiv preprint arXiv:1911.07596*, 2019.

J. Bergstra, D. Yamins, and D. Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123, 2013.

L. Bottou, F. E. Curtis, and J. Nocedal. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

A. Cauchy. Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538, 1847.

K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. *arXiv preprint arXiv:1812.02341*, 2018.

K. Cobbe, C. Hesse, J. Hilton, and J. Schulman. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019a.

K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman. Quantifying generalization in reinforcement learning. In *International Conference on Machine Learning*, pages 1282–1289, 2019b.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

H. Daneshmand, J. Kohler, A. Lucchi, and T. Hofmann. Escaping saddles with stochastic gradients. *arXiv preprint arXiv:1803.05999*, 2018.

D. Deutsch. *The beginning of infinity: Explanations that transform the world*. Penguin UK, 2011.

Y. Du, W. M. Czarnecki, S. M. Jayakumar, R. Pascanu, and B. Lakshminarayanan. Adapting auxiliary losses using gradient similarity. *arXiv preprint arXiv:1812.02224*, 2018.

A. E. Eshratifar, D. Eigen, and M. Pedram. Gradient agreement as an optimization objective for meta-learning. *arXiv preprint arXiv:1810.08178*, 2018.

V. Fomin, J. Anmol, S. Desroziers, J. Kriss, A. Tejani, and E. Rippeth. High-level library to help with training neural networks in pytorch. `https://github.com/pytorch/ignite`, 2020.

S. Fort, P. K. Nowak, and S. Narayanan. Stiffness: A new perspective on generalization in neural networks. *arXiv preprint arXiv:1901.09491*, 2019.

Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

S. Ghadimi and G. Lan. Stochastic first-and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 23(4):2341–2368, 2013.

T. Haavelmo. The statistical implications of a system of simultaneous equations. *Econometrica*, 11 (1), 1943.

C. Heinze-Deml and N. Meinshausen. Conditional variance penalties and domain shift robustness. *arXiv preprint arXiv:1710.11469*, 2017.

C. Heinze-Deml, J. Peters, and N. Meinshausen. Invariant causal prediction for nonlinear models. *Journal of Causal Inference*, 6(2), 2018.

L. Hurwicz. On the structural form of interdependent systems. In E. Nagel, P. Suppes, and A. Tarski, editors, *Logic, Methodology and Philosophy of Science, Proceedings of the 1960 International Congress*, pages 232–239. Stanford University Press, Stanford, CA, 1962.

S. Jastrzębski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

C. Jin, P. Netrapalli, and M. I. Jordan. Accelerated gradient descent escapes saddle points faster than gradient descent. *arXiv preprint arXiv:1711.10456*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

W. Krichene, A. Bayen, and P. L. Bartlett. Accelerated mirror descent in continuous and discrete time. In *Advances in neural information processing systems*, pages 2845–2853, 2015.

H. Kushner and G. G. Yin. *Stochastic approximation and recursive algorithms and applications*, volume 35. Springer Science & Business Media, 2003.

J. D. Lee, M. Simchowitz, M. I. Jordan, and B. Recht. Gradient descent converges to minimizers. *arXiv preprint arXiv:1602.04915*, 2016.

S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

J. Mooij, D. Janzing, J. Peters, and B. Schölkopf. Regression by dependence minimization and its application to causal inference in additive noise models. In *Proceedings of the 26th annual international conference on machine learning*, pages 745–752, 2009.

K. Muandet, D. Balduzzi, and B. Schölkopf. Domain generalization via invariant feature representation. In *International Conference on Machine Learning*, pages 10–18, 2013.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch, 2017.

J. Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2nd edition, 2009.

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

J. Quionero-Candela, M. Sugiyama, A. Schwaighofer, and N. D. Lawrence. *Dataset shift in machine learning*. The MIT Press, 2009.

M. Rojas-Carulla, B. Schölkopf, R. Turner, and J. Peters. Invariant models for causal transfer learning. *The Journal of Machine Learning Research*, 19(1):1309–1342, 2018.

L. Sagun, U. Evci, V. U. Guney, Y. Dauphin, and L. Bottou. Empirical analysis of the hessian of over-parametrized neural networks. *arXiv preprint arXiv:1706.04454*, 2017.

B. Schölkopf. Causality for machine learning, 2019. arXiv:1911.10500.

B. Schölkopf, D. Janzing, J. Peters, E. Sgouritsa, K. Zhang, and J. M. Mooij. On causal and anticausal learning. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, pages 1255–1262, 2012.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

H. A. Simon. Causal ordering and identifiability. In W. C. Hood and T. C. Koopmans, editors, *Studies in Econometric Methods*, pages 49–74. John Wiley & Sons, New York, NY, 1953. Cowles Commission for Research in Economics, Monograph No. 14.

A. Subbaswamy, P. Schulam, and S. Saria. Preventing failures due to dataset shift: Learning predictive models that transport. *arXiv preprint arXiv:1812.04597*, 2018.

M. Sugiyama and M. Kawanabe. *Machine learning in non-stationary environments: Introduction to covariate shift adaptation*. MIT press, 2012.

M. Sugiyama, M. Krauledat, and K.-R. Müller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8(May):985–1005, 2007.

V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., 1995. ISBN 0-387-94559-8.

J. von Kügelgen, A. Mey, and M. Loog. Semi-generative modelling: Covariate-shift adaptation with cause and effect features. *arXiv preprint arXiv:1807.07879*, 2018.

X. Wang, S. Ma, D. Goldfarb, and W. Liu. Stochastic quasi-newton methods for nonconvex stochastic optimization. *SIAM Journal on Optimization*, 27(2):927–956, 2017.

R. Ward, X. Wu, and L. Bottou. Adagrad stepsizes: Sharp convergence over nonconvex landscapes. In *International Conference on Machine Learning*, pages 6677–6686, 2019.

Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney. Pyhessian: Neural networks through the lens of the hessian. *arXiv preprint arXiv:1912.07145*, 2019.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

G. Zhang, L. Li, Z. Nado, J. Martens, S. Sachdeva, G. Dahl, C. Shallue, and R. B. Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, pages 8194–8205, 2019a.

Y. Zhang, W. Yu, and G. Turk. Learning novel policies for tasks. *ICML*, 2019b.

Appendix to Section 2

Section 2.1: A classic example of a patchwork solution

Consider a neural network with one hidden layer consisting of two neurons and sigmoidal activations:

$$f_\theta(x) = \theta_5\sigma(\theta_1 x + \theta_2) + \theta_6\sigma(\theta_3 x + \theta_4), \quad \sigma(z) := 1/(1+e^{-z}). \tag{5}$$

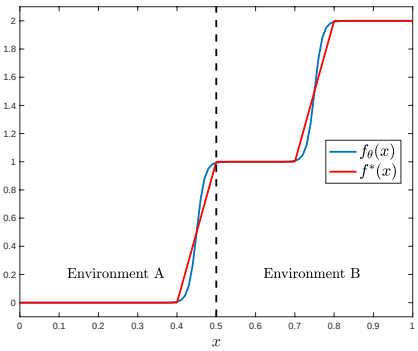
We want to learn the continuous function $f^* : [0,1] \to [0,2]$ defined as

$$f^*(x) = \begin{cases} 0 & x \in [0, 0.4); \\ 10(x-0.4) & x \in [0.4, 0.5); \\ 1 & x \in [0.5, 0.7); \\ 10(x-0.7)+1 & x \in [0.7, 0.8); \\ 2 & x \in [0.8, 1]. \end{cases}$$

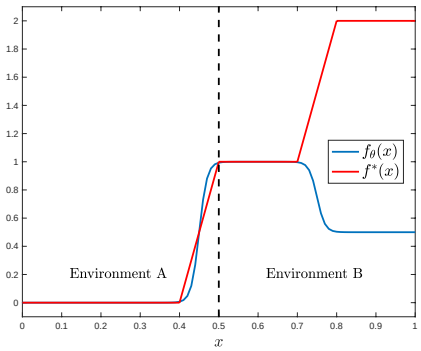
To perform this task, we have access to (noiseless) data from two environments:

$$A : \{(x, f(x)) \mid x \in [0, 0.5)\}, \quad B : \{(x, f(x)) \mid x \in [0.5, 1]\}.$$

There is a simple *constructive* way, provided by the universal function approximation theorem Cybenko [1989] to fit this function[15] using f_θ up to an arbitrarily small mean squared error $\mathcal{L}_{A+B}(\theta^*)$. Leaving out the details of such a construction (Cybenko [1989] for details), the reader can check on the left panel of Figure 11 that $\theta^* = (100, -50, 100, -75, 1, 1)$ provides a good fit for *both environments* A and B — both $\mathcal{L}_A(\theta^*)$ and $\mathcal{L}_B(\theta^*)$ are small.



$\theta = \theta^* = (100, -45, 100, -75, 1, 1)$



$\theta = \tilde{\theta}^* = (100, -45, 100, -75, 1, -0.5)$

Figure 11: Performance of the neural network in Equation 5 for two different parameters. Any reasonable modification on θ_6 (say ± 1) leaves the performance on environment A unchanged, while the performance on environment B quickly degrades.

However, it is easy to realize that θ^* — while being a solution which can be returned by gradient descent using the pooled data A+B — is not *consistent* (formal definition given in the main paper in Section 2). Indeed, it is possible to modify $\tilde{\theta}^*$ such that the loss in environment A remains almost unchanged, while the loss in environment B gets larger. In particular, on the right panel of Figure 11, we show that $\tilde{\theta}^* = (100, -50, 100, -75, 1, -0.5)$ is such that $\mathcal{L}_A(\theta^*) \leq \mathcal{L}_A(\tilde{\theta}^*) + \epsilon$ (with ϵ very small) but $\mathcal{L}_B(\theta^*) \ll \mathcal{L}_B(\tilde{\theta}^*)$. According to our definition in Equation 2 (see main paper), we have $C^\epsilon(\theta^*) \leq -|\mathcal{L}_B(\theta^*) - \mathcal{L}_B(\tilde{\theta}^*)|$ — that is a large negative number (low consistency).

Remark 1 (Connection to out of distribution generalization). The main point of this analysis was to show an example of where our measure of *consistency* behaves according to expectations: A typical implementation of the universal approximation theorem — which one would *not* expect to generalize out of distribution, due to its *'patchwork'* behavior — leads indeed to a very low consistency score.

[15] For a graphical description, the reader can check `http://neuralnetworksanddeeplearning.com/chap4.html`

Section 2.2: Consistency as arithmetic/geometric mean of landscapes

Here we show how the consistency score introduced in Equation 2 can be linked (in a simplified setting) to a comparison between the arithmetic and geometric means of the Hessians approximating the landscapes of two separate environments A and B.

At the local minimizer $\theta^* = 0$, we assume that $\mathcal{L}_A = \mathcal{L}_B = 0$ and consider the local quadratic approximations $\mathcal{L}_A(\theta) = \frac{1}{2}\theta^\top H_A \theta$ and $\mathcal{L}_B(\theta) = \frac{1}{2}\theta^\top H_B \theta$. Here, we make the additional simplifying assumption that H_A and H_B are diagonal (or, more broadly, co-diagonalizable): $H_A = \text{diag}(\lambda_1^A, \cdots, \lambda_n^A)$, $H_B = \text{diag}(\lambda_1^B, \cdots, \lambda_n^B)$, with $\lambda_i^A \geq 0$ and $\lambda_i^B \geq 0$ for all $i = 1, \ldots, n$. The *arithmetic* and *geometric* means (noted as H_{A+B} and $H_{A \wedge B}$) of these matrices are defined in this simplified setting as follows:

$$H_{A+B} = \text{diag}\left(\frac{1}{2}(\lambda_1^A + \lambda_1^B), \cdots, \frac{1}{2}(\lambda_n^A + \lambda_n^B)\right), \quad H_{A \wedge B} = \text{diag}\left(\sqrt{\lambda_1^A \lambda_1^B}, \cdots, \sqrt{\lambda_n^A \lambda_n^B}\right).$$

As motivated in the main paper and in Figure 12, one can link the consistency of two landscapes to a comparison between the geometric and arithmetic means of the corresponding Hessians.

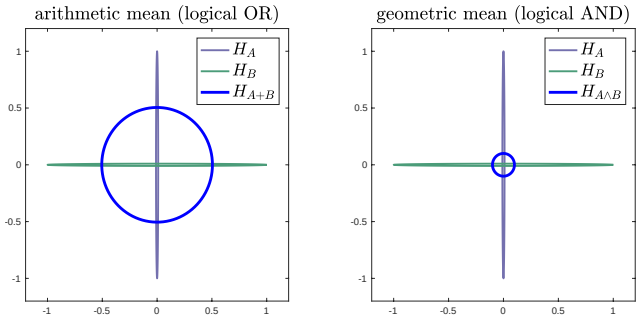


Figure 12: $H_A = \text{diag}(0.01, 1)$ and $H_B = \text{diag}(1, 0.01)$. The geometric average retains the volume of the original ellipses, while the volume of H_{A+B} is 100 times bigger. This magnification indicates that landscape A is not consistent with landscape B.

Proposition 3. In the setting we just described, the consistency score in Equation 2 can be estimated as follows:

$$C^\epsilon(\theta^*) \geq -2\epsilon \left(\frac{\det(H_{A+B})}{\det(H_{A \wedge B})}\right)^2.$$

Before showing the proof, we note that the proposition gives a *lower bound* on the consistency. That is, it provides a *pessimistic* estimate. Yet, as we motivated, this estimate has a nice geometric interpretation. However, as we outline in a remark after the proof, this estimate is tight in two important limit cases.

Proof. In this setting, Equation 2 gives

$$\mathcal{C}^\epsilon(\theta^*) := -\max\left\{\max_{\mathcal{L}_A(\theta) \leq \epsilon} \mathcal{L}_B(\theta), \max_{\mathcal{L}_B(\theta) \leq \epsilon} \mathcal{L}_A(\theta)\right\}.$$

Recall that

$$\mathcal{L}_A(\theta) = \frac{1}{2}\theta^\top H_A \theta = \frac{1}{2}\sum_i \lambda_i^A \theta_i^2.$$

Hence, this is a simple quadratic program with quadratic constraints, and

$$\max_{\mathcal{L}_A(\theta) \leq \epsilon} \mathcal{L}_B(\theta) = \max_{\frac{1}{2}\sum_i \lambda_i^A \theta_i^2 \leq \epsilon} \frac{1}{2}\sum_i \lambda_i^B \theta_i^2.$$

Further, we can change variables and introduce $\tilde{\theta}_i = \theta_i \sqrt{\lambda_i^A / 2}$. The problem gets even simpler:

$$\max_{\mathcal{L}_A(\theta) \leq \epsilon} \mathcal{L}_B(\theta) = \max_{\|\tilde{\theta}\|^2 \leq \epsilon} \sum_i \frac{\lambda_i^B}{\lambda_i^A} \tilde{\theta}_i^2 = \epsilon \cdot \max_i \frac{\lambda_i^B}{\lambda_i^A}.$$

All in all, we get

$$\begin{aligned}
-\mathcal{C}^\epsilon(\theta^*) &= \epsilon \max\left\{\max_i \frac{\lambda_i^B}{\lambda_i^A}, \max_i \frac{\lambda_i^A}{\lambda_i^B}\right\} \\
&= \epsilon \cdot \max_i \max\left\{\frac{\lambda_i^B}{\lambda_i^A}, \frac{\lambda_i^A}{\lambda_i^B}\right\} \\
&\leq \epsilon \cdot \max_i \left(\frac{\lambda_i^B}{\lambda_i^A} + \frac{\lambda_i^A}{\lambda_i^B}\right) \\
&= \epsilon \cdot \max_i \left\{\frac{(\lambda_i^B)^2 + (\lambda_i^A)^2}{\lambda_i^B \lambda_i^A}\right\} \\
&\leq \epsilon \cdot \max_i \left\{\frac{(\lambda_i^B + \lambda_i^A)^2}{\lambda_i^B \lambda_i^A}\right\}.
\end{aligned}$$

This means

$$\sqrt{-C^\epsilon(\theta^*)} \leq \epsilon \max_i \frac{\lambda_i^B + \lambda_i^A}{\sqrt{\lambda_i^B \lambda_i^A}} \leq 2\epsilon \frac{\prod_i (\lambda_i^B + \lambda_i^A)/2}{\prod_i \sqrt{\lambda_i^B \lambda_i^A}} = 2\epsilon \frac{\det(H_{A+B})}{\det(H_{A \wedge B})},$$

where the first inequality comes from the monotonicity of the square root function, and the second inequality comes from the fact that (i) the geometric mean is always smaller or equal than the arithmetic mean and (ii) for any sequence of numbers $\alpha_i > 1$, $\max_i \alpha_i \leq \prod_i \alpha_i$. □

Remark 2 (Sanity check). There are two important cases where we can test the bound above. First, if $H_A = H_B$, then $C^\epsilon(\theta^*) = -\epsilon$, and the bound returns $C^\epsilon(\theta^*) \geq -2\epsilon$, since the geometric and arithmetic mean are the same. Next, say $\lambda_i^A = 0$ but $\lambda_i^B > 0$; then, both the bound and the consistency score are $-\infty$ (lowest possible consistency).

Section 2.3: Proof of Proposition 1

In this appendix section we consider the AND-masked GD algorithm, introduced at the end of Section 2. We recall that the masked gradients at iteration k are $m_t(\theta^k) \odot \nabla\mathcal{L}(\theta^k)$, where $m_t(\theta^k)$ vanishes for any component where there are less than $t \in \{d/2+1, \ldots, d\}$ agreeing gradient signs across environments, and is equal to one otherwise. In a full-batch setting, the algorithm is

$$\theta^{k+1} = \theta^k - \eta\, m_t(\theta^k) \odot \nabla\mathcal{L}(\theta^k), \qquad \text{(AND-masked GD)}$$

where $\eta > 0$ is the learning rate.

Proposition 1. Let $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ have L-Lipschitz gradients and consider AND-masked GD with learning rate $\eta \leq 1/L$. After k iterations, AND-masked GD visits at least once a point θ where $\|m_t(\theta) \odot \nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(1/k)$.

Proof. Thanks to the component-wise L-smoothness and using a Taylor expansion around θ^i we have

$$\begin{aligned}
\mathcal{L}(\theta^{i+1}) &\leq \mathcal{L}(\theta^i) - \eta\langle\nabla\mathcal{L}(\theta^i), m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\rangle + \frac{L\eta^2}{2}\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \\
&= \mathcal{L}(\theta^i) - \left(\eta - \frac{L\eta^2}{2}\right)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2.
\end{aligned}$$

If we seek $\eta - L\eta^2/2 \geq \eta/2$, then $\eta \leq \frac{1}{L}$, as we assumed in the proposition statement. Therefore, $\mathcal{L}(\theta^{i+1}) \leq \mathcal{L}(\theta^i) - (\eta/2)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2$, for all $i \geq 0$. Summing over i from 0 to a desired iteration k, we get

$$\sum_{i=0}^{k-1} (\eta/2)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \leq \mathcal{L}(\theta^0) - \mathcal{L}(\theta^k) \leq \mathcal{L}(\theta^0).$$

Therefore,

$$\min_{i=0,\ldots,k} \|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \leq \frac{1}{k}\sum_{i=0}^{k-1} (\eta/2)\|m_t(\theta^i) \odot \nabla\mathcal{L}(\theta^i)\|^2 \leq \frac{2\mathcal{L}(\theta^0)}{\eta k}.$$

Hence, there exist an iteration $i^* \in \{0, \ldots, k\}$ such that $\|m_t(\theta^{i^*}) \odot \nabla\mathcal{L}(\theta^{i^*})\|^2 \leq \mathcal{O}(1/k)$. □

Section 2.3: Proof of Proposition 2

Here we fix parameters $\theta \in \mathbb{R}^n$ and assume gradients $\nabla\mathcal{L}_e(\theta) \in \mathbb{R}^n$ coming from environments $e \in \mathcal{E}$ are drawn independently from a multivariate Gaussian with zero mean and $\sigma^2 I$ covariance. We want to show that, in this random setting, the AND-mask introduced in Section 2.3 decreases the magnitude of the gradient step.

Proposition 2. Consider the setting we just outlined, with $\mathcal{L} = (1/|\mathcal{E}|)\sum_{e\in\mathcal{E}}\mathcal{L}_e$ and $d := |\mathcal{E}|$. While $\mathbb{E}\|\nabla\mathcal{L}(\theta)\|^2 = \mathcal{O}(n/d)$, we have that for any $t \in \{d/2+1, \dots, d\}$ there exists $1 < c \leq 2$ such that $\mathbb{E}\|m_t(\theta)\odot\nabla\mathcal{L}(\theta)\|^2 \leq \mathcal{O}(n/c^d)$.

Proof. Let us drop the argument θ for ease of notation. First, let us consider $\nabla\mathcal{L}$ (no gradient AND-mask):

$$\mathbb{E}\left\|\frac{1}{d}\sum_{i=1}^{d}\nabla\mathcal{L}_{e_i}\right\|^2 = \frac{1}{d^2}\sum_{i=1}^{d}\mathbb{E}\|\nabla\mathcal{L}_{e_i}\|^2 = \frac{n\sigma^2}{d},$$

where in the first equality we used the fact that the $\nabla\mathcal{L}_{e_i}$ are uncorrelated and in the second the fact that $\mathbb{E}[\|\nabla\mathcal{L}_{e_i}\|^2]$ is the trace of the covariance of $\nabla\mathcal{L}_{e_i}$.

Next, assume we apply the element-wise AND-mask m_t to the gradients, which puts to zero the components (dimensions) where there are less than $t \in \{d/2, \dots, d\}$ equal signs. Since Gaussians are symmetric around zero, the probability of having *exactly* u positive j-th gradient component among d environments is $Pr(p_j = u) = \left(\frac{1}{2}\right)^d\binom{d}{u}$. Hence, the probability to keep the j-th gradient direction (considering also negative consistency) is

$$\begin{aligned}\Pr[[m_t]_j = 1] &= \sum_{u=t}^{d}\Pr(p_j = u) + \sum_{u=0}^{d-t}\Pr(p_j = u)\\ &= \left(\frac{1}{2}\right)^d\sum_{k=t}^{d}\binom{d}{k} + \left(\frac{1}{2}\right)^d\sum_{k=0}^{d-t}\binom{d}{k}\\ &= 2\left(\frac{1}{2}\right)^d\sum_{k=t}^{d}\binom{d}{k}. \end{aligned} \tag{6}$$

We would now like to compute $\mathbb{E}\left\|m_t \odot \left(\frac{1}{d}\sum_{i=1}^{d}\nabla\mathcal{L}_{e_i}\right)\right\|^2$. The difficulty lies in the fact that the event $m_t = 1$ makes gradients conditionally *dependent*. Indeed, conditioning on both $m_t = 1$ and $[\nabla\mathcal{L}_e]_j > 0$ changes the distribution of $[\nabla\mathcal{L}_{e'}]_j$: this gradient entry is going to be more likely to be positive or negative, depending on the value of $[\nabla\mathcal{L}_e]_j$ and on the details of the gradient mask. To solve the issue, we our strategy is to reduce the discussion (without loss in generality and with no additional assumption) to the case where gradient entries have all the same sign and hence conditional independence is restored.

We consider the following writing for the quantity we are interested in:

$$\begin{aligned}\mathbb{E}\left\|m_t \odot \left(\frac{1}{d}\sum_{i=1}^{d}\nabla\mathcal{L}_{e_i}\right)\right\|^2 &= \sum_{j=1}^{n}\mathbb{E}\left[[m_t]_j\left(\frac{1}{d}\sum_{i=1}^{d}[\nabla\mathcal{L}_{e_i}]_j\right)^2\right]\\ &= \sum_{j=1}^{n}\sum_{\hat{p}_j=0}^{d}\mathbb{E}\left[[m_t]_j\left(\frac{1}{d}\sum_{i=1}^{d}[\nabla\mathcal{L}_{e_i}]_j\right)^2\middle| p_j = \hat{p}_j\right]\Pr[p_j = \hat{p}_j]\\ &= \sum_{j=1}^{n}\sum_{\hat{p}_j=0}^{(d-t)}\sum_{\hat{p}_j=t}^{d}\mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d}[\nabla\mathcal{L}_{e_i}]_j\right)^2\middle| p_j = \hat{p}_j\right]\Pr[p_j = \hat{p}_j]\\ &= 2\sum_{j=1}^{n}\sum_{\hat{p}_j=t}^{d}\mathbb{E}\left[\left(\frac{1}{d}\sum_{i=1}^{d}[\nabla\mathcal{L}_{e_i}]_j\right)^2\middle| p_j = \hat{p}_j\right]\left(\frac{1}{2}\right)^d\binom{d}{\hat{p}_j},\end{aligned}$$

where we used the definition of 2-norm, the law of total expectation, and the symmetry of the problem with respect to positive and negative numbers. Finally, since the gradient components within the same environment are conditionally independent, for any $j \in \{1, \dots, n\}$ we can write

$$\mathbb{E}\left\| m_t \odot \left(\frac{1}{d} \sum_{i=1}^{d} \nabla \mathcal{L}_{e_i} \right) \right\|^2 = 2n \sum_{\hat{p}_j = t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \right] \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j}.$$

Finally, we note that the following bound holds:

$$\mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = \hat{p}_j \leq d \right] \leq \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = d \right].$$

Indeed, if *all* environments lead to positive (or, symmetrically, negative) and *non-interacting* gradients in the j-th direction, the average will be the biggest in norm. Moreover — crucially — conditioned on the event $p_j = d$, gradients coming from different environments are distributed as a positive half-normal distributions. Moreover, they are *conditionally independent*; this because, since they are all positive, the value of a gradient in one environment cannot influence the value of the gradient in another one. We remark that conditional independence on the right-hand side is therefore *not an assumption*, but is intrinsic to the upper bound.

Putting it all together, we have

$$\begin{aligned}
\mathbb{E}\left\| m_t \odot \left(\frac{1}{d} \sum_{i=1}^{d} \nabla \mathcal{L}_{e_i} \right) \right\|^2 &\leq 2n \sum_{\hat{p}_j = t}^{d} \mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} [\nabla \mathcal{L}_{e_i}]_j \right)^2 \middle| p_j = d \right] \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j} \\
&\leq 2n \sum_{\hat{p}_j = t}^{d} \sigma^2 \left(\frac{1}{2}\right)^d \binom{d}{\hat{p}_j} \\
&\leq \sigma^2 n (d-t) \binom{d}{t} \left(\frac{1}{2}\right)^{d-1},
\end{aligned}$$

where in the second line we bounded the squared average of a sum of half normal distributions: let $\{X_i\}_{i=1}^d$ be a family of uncorrelated positive half-normal distributions derived from a Gaussians with mean zero and variance σ^2, we have[16] that $\mathbb{E}[X_i] = \sigma\sqrt{2/\pi}$ and $\mathbb{E}[X_i^2] = \sigma^2$. Also, $\mathbb{E}[X_i X_j] = \mathbb{E}[X_i]\mathbb{E}[X_j] \leq \sigma^2$. Therefore,

$$\mathbb{E}\left[\left(\frac{1}{d} \sum_{i=1}^{d} X_i \right)^2 \right] = \frac{1}{d^2} \sum_{i,j=1}^{d} \mathbb{E}[X_i X_j] \leq \sigma^2.$$

Finally, if we set $r = t/d \in (0.5, 1]$, we have[17]

$$\binom{d}{t} \sim \left(\frac{1}{r^r (1-r)^{1-r}} \right)^d$$

as $d \to \infty$ (discarding all polynomial terms). Hence $\binom{d}{t}$ is of the form q^d, with $1 \leq q < 2$. So, the quantity $\sigma^2 n(d-t)\binom{d}{t}\left(\frac{1}{2}\right)^{d-1}$ will be exponentially decreasing at a rate $\mathcal{O}(n/(2-q)^d)$. Notably, if $t = d/2$, then we lose the exponential rate and get back to $\mathcal{O}(n/d)$. □

Appendix to Section 3

Section 3.1

The code for the dataset and all experiments presented in this paper is released with the supplementary materials.

[16] `https://en.wikipedia.org/wiki/Half-normal_distribution`

[17] Theorem 1 in Burić, Tomislav, and Neven Elezović. "Asymptotic expansions of the binomial coefficients." Journal of applied mathematics and computing 46.1-2 (2014): 135-145.

Table 1: Hyperparameter ranges for synthetic data experiments. The regularizers L1 and L2 are never combined; instead, one weight regularization type out of L1, L2 and none is selected and we sample from the respective range afterwards.

Hyperparameter	Ranges
No. hidden units	$\{256, 512\}$
No. hidden layers	$\{3, 5\}$
Batch-size	$\{64, 128, 256\}$
Optimizer	$\{\text{Adam}_{\beta_1=0.9,\beta_2=0.999}$, SGD + $\text{momentum}_{0.9}\}$
Learning rate	{1e-3, 1e-2, 1e-1}
Batch-normalization	{Yes, No}
Dropout	$\{0.0, 0.5\}$
L2 regularization	{1e-5, 1e-4, 1e-3}
L1 regularization	{1e-6, 1e-5, 1e-4}

Dataset

Here we report more technical details about the synthetic dataset described in Section 3.

Each example is constructed as follows: we first choose the label randomly to be either $+1$ or -1, with equal probability. The example is a vector with $d_S + d_M$ entries, consisting of the *shortcut* and the *mechanism*. In our experiments, $d_M = 2$ and $d_S = 32$.

The Gaussian *shortcuts* are obtained by first sampling one random vector $\mathbf{x}_s \in \mathbb{R}^{d_S}$ per environment. Its components $x_{s,i}$ are sampled independently from a Normal distribution: $x_{s,i} \sim \mathcal{N}(0, 0.1)$. We use $\mathbf{x}_s$ for class 1, and $-\mathbf{x}_s$ for class -1. In the test set, all shortcut components are sampled i.i.d. from the same Normal distribution. Effectively, each example of the test set belongs to a different domain. The *mechanism* is implemented as the two interconnected spirals shown in Figure 13 by sampling the radius $r \sim \text{Unif}(0.08, 1.0)$ and then computing the angle as $\alpha = 2\pi nr$ where n is the number of revolutions of the spiral. We add uniform noise in the range $[-0.02, 0.02]$ to the radii afterwards.

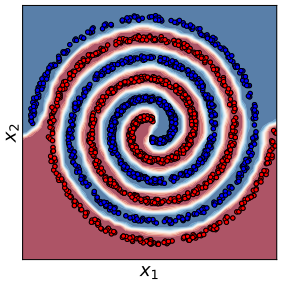


Figure 13: The spirals used as the *mechanism* in the synthetic memorization dataset.

The training dataset consists of 1280 examples per environment and we use $D = 32$ environments unless otherwise mentioned. The training datasets consists of 2000 examples.

Experiment

We train all networks for $\lfloor 3000/D \rfloor$ epochs, dropping the learning rate by a factor 10 halfway through, and again at three-quarters of training. For computational reason, we stop each trial before completion if the training accuracy exceeds 97% and the test accuracy is below 60%. All networks are MLPs with LeakyReLU activation functions and a cross-entropy loss on the output. We run a hyperparameter search over the ranges shown in Table 1. For IRM and the AND-mask, we select the best-performing run and re-run it 50 times with different random seeds. For DANN and the standard baselines nothing produced results significantly better than chance.

Standard regularizers and AND-mask

The networks with the L1, L2, Dropout and Batch-normalization regularizers, have hyperparameters that were randomly selected from Table 1. For the AND-mask we used the very same ranges. The regularizers L1 and L2 are never combined; instead, one weight regularization type out of L1, L2 and none is selected and we sample from the respective range afterwards.

Domain Adversarial Neural Networks

The experiments using DANN follow a similar pattern. The model consists of an embedding network, a classification network, and a "domain discrimination" network. All three modules are two-layer multi-layer perceptrons (MLP). The number of hidden units of all MLPs are sampled from the range specified in Table 1, and we trained 100 models. Both label classifier and domain discriminator are applied to the output of the embedding network. The label classifier is trained to minimize the cross-entropy-loss between the predicted and the true label. Similarly, the domain discriminator is trained to minimize the loss between predicted and true domain-label. The embedding network is trained to minimize the regular task classification loss and at the same time to maximize the the domain-loss achieved by the domain discriminator.

Invariant Risk Minimization

For the experiments using IRM we used the authors' PyTorch implementation from `https://github.com/facebookresearch/InvariantRiskMinimization`. We perform a random hyperparameter search over with the ranges shown in Table 2

Table 2: Hyperparameter ranges for IRM.

Hyperparameter	Ranges
No. hidden units	$\{256, 512\}$
No. hidden layers	$\{3, 5\}$
Batch-size	$\{64, 128, 256\}$
Optimizer	$\{\text{Adam}_{\beta_1=0.9,\beta_2=0.999}$, SGD + $\text{momentum}_{0.9}\}$
Batch-normalization	{Yes, No}
Penalty weight	$\{10.0, 100.0, 1000.0\}$
Number of annealing iterations	$\{0, 1, 2, 4, 8\}$
Learning rate	{1e-3, 1e-2, 1e-1, 1}

Curves for all experiments

In Figure 14 we show the learning curves of training and test accuracy for the different methods.

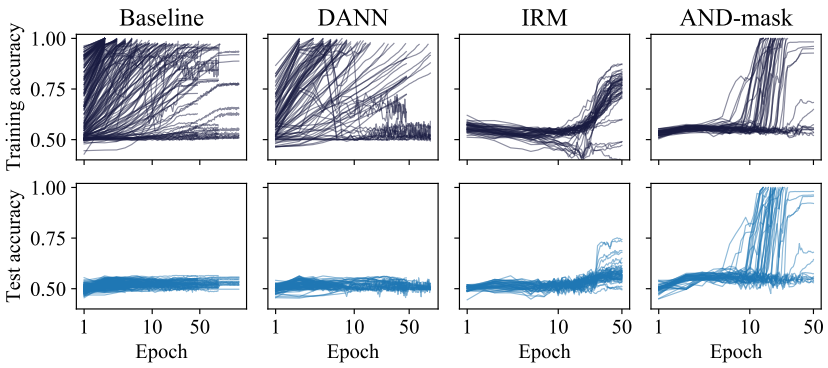


Figure 14: Learning curves for the evaluated methods. The top row shows the accuracy on the training set, the bottom row shows the accuracy on the test set.

Correlation plots

For the correlation plots in Figure 7 we used a randomly initialized MLP with the following configuration: 3 hidden layers, 256 hidden units. The dataset was using 16 environments and batches of size 1024. The lines in Figure 7 are linear least-squares regressions to the gradient data shown as scatter plots. We repeat the experiment 10 times with different network weight seeds, resulting in the 10 regression lines. Zero gradients are excluded from the regression computation, as most gradients are masked out by the product mask in both cases.

Further visualizations and experiments

In Figure 15 we show how many environments need to be present for the baseline without AND-mask to switch the decision boundary from the shortcuts to the mechanism. Under the same experimental condition as in the main paper, the baseline first succeeds at 1024 environments.

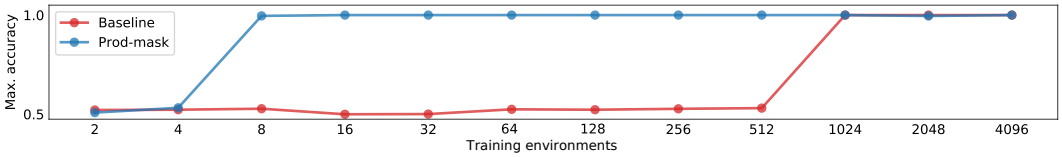


Figure 15: Relationship between number of training environments and test accuracy for the AND-mask method compared to the baseline. We show the best performance out of five runs using the settings that were used for the experiment in the main text.

Section 3.2: CIFAR-10 memorization and label noise experiments

Memorization experiment In Figure 16, we report the test performance (dashed lines) corresponding to the curves presented in the main paper for the CIFAR-10 memorization experiment. The test performance with standard labels decreases slower than the training performance as the threshold increases, and they eventually reach the same value. This is consistent with the hypothesis that by training on the consistent directions, the AND-mask selects the invariant patterns and prunes out the signals that are not invariant.

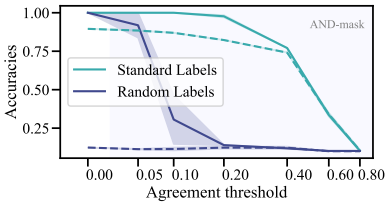


Figure 16: Dashed lines show test acc, solid lines show training acc.

Network architecture and training details Each trial trains the ResNet "`FastResNet`" from the PyTorch-Ignite example[18] for 80 epochs on the full CIFAR-10 training set. We use the Adam optimizer with a learning rate of $5\text{e}{-4}$, and a 0.1 learning rate decay at epoch 40 and 60. We fix the batch size to 80. We set up 14 trials by evaluating each of the AND-mask-thresholds $\{0, 0.05, 0.1, 0.2, 0.4, 0.6, 0.8\}$ for two datasets: (a) unchanged CIFAR-10, (b) CIFAR-10 with the training labels replaced by random labels. Note that a threshold of 0 corresponds to not using the AND-mask. Each trial is run twice with separate random seeds.

Label noise experiment We trained the same ResNet as for the experiment above, once with and once without the AND-mask. We ran each experiment with three different starting learning rates $\{5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}\}$ and a learning rate decay at epoch 60. The baseline worked best with a learning rate of $1\text{e}{-3}$, while the AND-mask with $5\text{e}{-3}$, likely to compensate for the masked out gradients. The AND-mask threshold that worked best was 0.2, which is consistent with the results obtain in the experiment above.

Section 3.3: Behavioral Cloning on CoinRun

The target policy π^* is obtained by training PPO [Schulman et al., 2017] for 400M time steps using the code[19] for the paper Cobbe et al. [2019a]. This policy is trained on the full distribution of levels in order to maximize its generality. We use π^* to generate a behavioral cloning (BC) dataset, consisting of pairs $(s, \pi^*(a|s))$, where s are the input-images (64×64 RGB) and $\pi^*(a|s)$ is the discrete probability distribution over actions output by π^*.

The states are sampled randomly from trajectories generated by π^*. In order to test for generalization performance, the BC training dataset is restricted to 64 distinct levels. We generate 1000 examples per training level. The test set consists of 2000 examples, each from a different level which does not appear in the training set.

[18] `https://github.com/pytorch/ignite/blob/master/examples/contrib/cifar10/fastresnet.py`

[19] `https://github.com/openai/train-procgen`

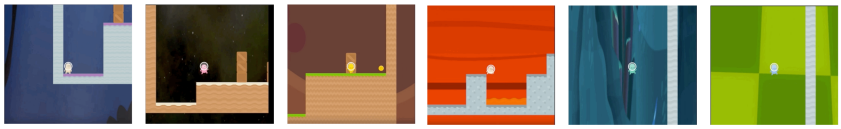


Figure 17: Screenshots of 6 levels of CoinRun (from OpenAI).

A ResNet-18 $\hat{\pi}_\theta$ is trained to minimize the loss $D_{\mathrm{KL}}(\pi^*||\hat{\pi}_\theta)$. We ran two automatic hyperparameter optimization studies using Tree-structured Parzen Estimation (TPE) [Bergstra et al., 2013] of 1024 trials each, with and without the AND-mask. The learning rate was decayed by a factor of 10 half-way at at $3/4$ of the training epochs.

The "temporal" version of the AND-mask used for this experiment is reported in Algorithm 1.

Algorithm 1: Temporal AND-mask Adam

$\mathbf{m} \leftarrow \beta_1 \cdot \mathbf{m} + (1 - \beta_1) \cdot \mathbf{g}$

$\mathbf{v} \leftarrow \beta_2 \cdot \mathbf{v} + (1 - \beta_2) \cdot (\mathbf{g} \circ \mathbf{g})$

$\mathbf{a} \leftarrow \beta_3 \cdot \mathbf{a} + (1 - \beta_3) \cdot \texttt{elemwise\_sign}(\mathbf{g})$

$\mathbf{b} \leftarrow \mathbb{1}[|\mathbf{a}| \geq \tau]$

$\theta \leftarrow \theta - \alpha(\mathbf{m} \circ \mathbf{b}) \oslash \sqrt{\mathbf{v} + \epsilon}$

In blue we highlight the additional lines compared to traditional Adam. The threshold τ and β_3 are hyperparameters that we included in the 1'024 trials of the search using Tree-structured Parsen Estimators. For the top 10 runs, hyperparameter values that were selected via the TPE search for the AND-mask are the following.

Table 3: Hyperparameters for the 5 best runs using the AND-mask, from the TPE search.

Test KL div	lr	β_1	β_3	τ	weight decay
1.652e-2	0.0078	0.21	0.79	0.36	0.057
1.656e-2	0.0072	0.26	0.86	0.40	0.041
1.662e-2	0.0080	0.23	0.84	0.41	0.045
1.665e-2	0.0068	0.33	0.72	0.47	0.077
1.672e-2	0.0063	0.67	0.65	0.47	0.080

We found that applying weight decay as a second independent update *after* the AND-mask routine improved performance. To keep the comparison fair, we added this as a switch in the hyperparameter search for the Adam baseline as well, and it improved performance there as well.

Appendix to Section 4

Here we are going to compare ILC to other approaches for learning invariances in the data with neural networks, and in particular to Invariant Risk Minimization (IRM) Arjovsky et al. [2019]. The authors of IRM analyze a set up where minimizing training error might lead to models which absorb all the correlations found within the training data, thus failing to recover the relevant causal explanation. They consider a multi-environment setting and focus on the objective of extracting data representations that lead to invariant prediction across environments.

While the high level objective is close to the one we focused on, the differences become clear when considering the definition of *invariant predictors* presented in Arjovsky et al. [2019]:

Definition 1. A data representation $\Phi : \mathcal{X} \to \mathcal{H}$ elicits an invariant predictor $w \circ \Phi$ across environments $\mathcal{E}$ if there is a classifier $w : \mathcal{H} \to \mathcal{Y}$ simultaneously optimal for all environments, i.e., $w \in \arg\min_{\bar{w}:\mathcal{H}\to y} R^e(\bar{w} \circ \Phi)\ \forall e \in \mathcal{E}$.

In particular, the objective minimized by IRM is:

$$\min_{\Phi:\mathcal{X}\to\mathcal{Y}} \sum_{e\in\mathcal{E}_{\mathrm{tr}}} R^e(\Phi) + \lambda \cdot \left\|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\right\|^2 \tag{7}$$

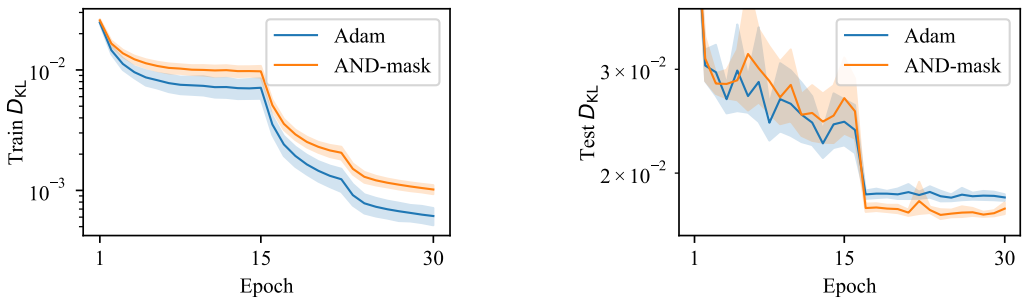


Figure 18: Learning curves for the behavioral cloning experiment on CoinRun. Training loss is shown on the left, test loss is shown on the right. We show the mean over the top-10 runs for each method. The shaded regions correspond to the 95% confidence interval of the mean based on bootstrapping.

where Φ are the logits predicted by the neural network and w is a dummy scaling variable (see Arjovsky et al. [2019]). The relevant part is the penalty term $\lambda \cdot \left\|\nabla_{w|w=1.0} R^e(w \cdot \Phi)\right\|^2$: One way to interpret it, is that the penalty is large on every environment where the distribution outputted by Φ could be made 'closer' to the distribution of the labels by either sharpening ($w > 1$) or softening it (i.e., closer to uniform $w < 1$).

Let us consider the example from IRM, where the authors describe two datasets of images that each contain either a cow or a camel: In one of the datasets, there is grass on 80% of the images with cows, while in the other dataset there is grass on 90% of them. IRM then makes the point that we can learn to ignore grass as a feature, because its correlation with the label cow is inconsistent (80% vs 90%). The setting we consider in this paper is slightly different: take our example from the CIFAR-10 experiments. Under our concept of invariance, we expect that (depending on the data generating process) even a single dataset where we treat every image as coming from its own 'environment' should be sufficient to discover invariances. Drawing a connection to the setting from IRM, we would argue that the second dataset should not be necessary to learn that 'grass' is not 'cow'. If one treats every example as coming from its own environment, there is already sufficient information in the first dataset to realize that cows are not grass: Grass is predictive of cows only in 80% of the data, so grass cannot be 'cow'. The actual cow on the other hand, should be present in 100% of the images, and as such it is the invariance we are looking for. Note that this is of course a much more strict definition of invariance: If our dataset contains images labeled as 'cows' but that have no cows within them, we might start to discard the features of cows as well.