SHARPNESS-AWARE MINIMIZATION ENHANCES FEATURE QUALITY VIA BALANCED LEARNING

Jacob Mitchell Springer[1], Vaishnavh Nagarajan[2] & Aditi Raghunathan[1]
[1]Carnegie Mellon University [2]Google Research

ABSTRACT

Sharpness-Aware Minimization (SAM) has emerged as a promising alternative to stochastic gradient descent (SGD) as an optimizer for neural networks. The originally-proposed motivation behind SAM was to bias models towards flatter minima that are believed to generalize better. However, recent studies have shown conflicting evidence on the relationship between flatness and generalization, suggesting that flatness does fully explain SAM's success. Sidestepping this debate, we identify an orthogonal effect of SAM that is beneficial out-of-distribution: we argue that SAM implicitly balances features. At a high level, we demonstrate that SAM improves the quality of features by adaptively suppressing well-learned features which gives remaining features opportunity to be learned. We show that this mechanism is beneficial in datasets that contain redundant or spurious features where SGD falls for the simplicity bias and would not otherwise learn all available features. Our insights are supported by experiments on real data: we demonstrate that SAM improves the quality of features in datasets containing redundant or spurious features, including CelebA, Waterbirds, CIFAR-MNIST, and DomainBed.

1 INTRODUCTION

Sharpness-Aware Minimization (SAM) has emerged as a compelling alternative to stochastic gradient descent (SGD) as an optimizer for neural networks (Bahri et al., 2021). The core motivation of SAM is to not just to minimize the original training objective, but to also find a minimum with small Hessian norm (Wen et al., 2022). This motivation has been inspired by conventional wisdom that "flat" minima may correspond to better in-distribution generalization in comparison to sharper minima (Keskar et al., 2016; Jiang et al., 2019) (Foret et al., 2020). However, there has been a long-standing debate on whether this flatness is even a reliable predictor of generalization (Dinh et al., 2017; Kaddour et al., 2022; Andriushchenko et al., 2023b; Wen et al., 2023; Kaur et al., 2023). To this end, alternative theories about SAM have been put forth to explain its performance gains, typically by studying specific inductive biases of SAM in simple theoretical settings (Andriushchenko & Flammarion, 2022; Andriushchenko et al., 2023a; Behdin & Mazumder, 2023).

In this paper, we provide a complementary perspective to the discussion relating flatness to SAM's in-distribution gains. We argue that SAM can lead to certain *out-of-distribution* gains via a *feature diversifying effect*. Specifically, we consider datasets with multiple *redundant* features (e.g., an image of a cat may be recognized by its fur or eyes or tail, or of a truck by its wheels or its front or side doors). In such datasets, SGD is known to suffer from a "simplicity bias" towards learning only the simplest features that suffice for the task, over a range of datasets (Shah et al., 2020). However, we show that SAM overcomes this bias to learn multiple highly correlated features. Remarkably, this happens even though the SAM objective was not explicitly designed with this anti-simplicity, diversity bias. This bias makes SAM favorable in downstream tasks with distributed shifts where a diverse set of features become relevant for prediction.

Why does SAM lead to this feature-diversifying effect, even when learning only one of the features would suffice to minimize the objective? We construct a minimal toy setting with a simple and a complex feature to address this question. In the toy setting, SGD fails to capture an informative representation of the complex feature due to the simplicity bias. By contrast, SAM is able to learn a high-quality representation of both the simple and the complex feature. We argue that the key difference between the two algorithms can be seen directly in their respective gradient update steps. By decomposing the SAM gradient update step into two terms, we are able to identify two separate effects of SAM that lead to the feature-diversifying effect. The first effect is a balancing of the

weight of the training examples during each update step, leading to a more uniform update across the dataset. This effect is similar to other methods that explicitly re-weight examples to promote complex-feature learning such as GroupDRO (Sagawa et al., 2019) and Just Train Twice (Liu et al., 2021). Unlike these methods, SAM re-weights points implicitly through its update step. The second effect is a balancing of the effective step size taken to learn each feature: the effective learning rates of well-learned features are suppressed. These effects lead to balanced learning of all features, and thus a more diverse representation. SGD, by contrast, does not exhibit these balancing effects.

The feature diversity induced by SAM is particularly important in the setting in which the features which are redundant in the training distribution are not all necessarily predictive of the label in a downstream distribution. For example, the Waterbirds dataset (Sagawa et al., 2019) consists of images of water birds and land birds on backgrounds of water and land. In the training distribution, the background and bird type are highly correlated, but downstream, the background is not predictive of the bird type and vice versa. Ideally, we learn a representation amenable to adaptation to predict bird type, or background. We find that SAM indeed improves the performance of transfer learning with last-layer retraining on the downstream Waterbirds tasks, and on other similar datasets, include CelebA, CIFAR-MNIST, and DomainBed.

Our perspective is complementary to the large body of existing literature that discusses the in-distribution improvements of SAM and variants (Foret et al., 2020; Na et al., 2022; Zhang et al., 2023; Zhao et al., 2022; Rangwani et al., 2022; Wang et al., 2022; Meng et al., 2023). Importantly, this finding allows us to contextualize the small set of prior work that has discovered that SAM helps out-of-distribution performance (Wang et al., 2023; Cha et al., 2021; Bahri et al., 2021). We hope that this work will inspire future work to further understand the mechanisms of SAM and will motivate new algorithms for improving downstream performance on out-of-distribution data in similar contexts.

In summary, our contributions are as follows:

1. We identify a feature diversifying effect of SAM in settings in which data has multiple redundant features, and verify that this effect improves feature representation quality.

2. We construct and analyze a minimal toy setting consisting of multiple redundant features where SGD fails to capture an informative representation of all features, but SAM succeeds.

3. We demonstrate that the feature diversifying effect of SAM arises from two interpretable balancing effects: a balancing effect that reweights the training examples to be more uniform and a balancing effect that suppresses the effective learning rate of well-learned features.

4. We verify that the improved feature representations can be used to improve performance on out-of-distribution downstream tasks in realistic settings.

2 Related work

Understanding Sharpness-Aware Minimization. The success of SAM is commonly explained by the connection between the flatness of the landscape and improved generalization (see Section 5 of Foret et al. (2020)). The connection continues to inspire flatness-optimizing objectives for a wide range of tasks (Abbas et al., 2022; Na et al., 2022; Wang et al., 2023; Zhang et al., 2023; Zhao et al., 2022; Rangwani et al., 2022; Wang et al., 2022; Meng et al., 2023).

However, recent work has questioned whether sharpness and generalization are as linked as previously thought. On the one hand, it appears that sharper minima can generalize well after all, and on the other, even flat minima may sometimes generalize poorly (Dinh et al., 2017; Kaddour et al., 2022; Andriushchenko et al., 2023b; Wen et al., 2023; Kaur et al., 2023). In this backdrop, some works on understanding SAM have departed from flatness. These explicitly characterize the implicit bias of SAM in simple theoretical settings to show that it can improve in-distribution generalization (Andriushchenko & Flammarion, 2022; Behdin & Mazumder, 2023). Our work takes a different approach departing from both flatness and deriving the implicit regularization by looking at how the SAM perturbation changes each gradient step along the trajectory. Further, while these works focus on in-distribution generalization, we focus on the feature-diversifying effect of SAM in out-of-distribution and transfer learning settings.

Benefits of SAM beyond in-distribution generalization. SAM has been shown to improve performance in across various other settings such as meta learning (Abbas et al., 2022), domain generalization (Wang et al., 2023; Cha et al., 2021), label noise (Foret et al., 2020), transfer learning

in language models (Bahri et al., 2021). Our work augments these findings by offering one unifying factor that can explain SAM's gains beyond in-distribution generalization—SAM learns higher quality representations of hard-to-learn features.

Feature learning in the presence of multiple predictive features. It has been shown that neural networks learn some features more easily than others, a tendency that is dubbed as a "simplicity bias" (Shah et al., 2020; Kalimeris et al., 2019; Morwani et al., 2023; Rahaman et al., 2019). Besides different learning speeds, the presence of one feature also impacts the *quality* (i.e. probing error) of another feature in the learned representation as shown in (Pezeshki et al., 2021). This has inspired various fixes such as regularizers (Pezeshki et al., 2021), selecting freezing of parameters (Ye et al., 2023) and data augmentation (Plumb et al., 2021). This problem has also been addressed in a line of work aiming to improve worst-group error in the presence of spurious correlations, notably Sagawa et al. (2019), but we defer a detailed discussion to Appendix A. In contrast to these algorithms that are explicitly designed to address the simplicity bias, we show that SAM does so *without being explicitly incentivized to do so*.

We elaborate on further related work including the broader literature on shortcut learning and spurious correlations, feature diversity and finetuning, and variants of the SAM algorithm in Appendix A.

3 Setup and preliminaries

Task. We consider the setting of classification where we map input $x \in \mathcal{X}$ to output $y \in \mathcal{Y}$. Given training data $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$ where $x_i, y_i \sim D_{\text{train}}$, our goal is to learn a neural network classifier $f : \mathcal{X} \mapsto \mathcal{Y}$. We are interested in the diversity of the representations used by a classifier. Hence it is convenient to parameterize the neural network classifiers f as linear classifiers on top of feature representations, i.e. $f(v, \theta; x) = \arg\max(v^\top \Phi_\theta(x))$, where $v \in \mathbb{R}^{k \times |\mathcal{Y}|}$ and $\Phi_\theta : \mathbb{R}^d \mapsto \mathbb{R}^k$. Note that the feature map Φ_θ is itself a neural network. We use $w = (v, \theta)$ to denote the weights of the network when we do not need to discuss the features explicitly.

Multiple predictive features. In our setting, there are several predictive features. As an example, inspired by Sagawa et al. (2019), we consider the CelebA inputs as having two redundant features: a gender feature {*male, female*} and hair color {*blond, dark*}, both, in part, predictive of the label. Formally, each input x has a set of features $a_1(x), a_2(x), \ldots a_m(x)$ where each $a_i(x) \in \mathcal{A}_i$ denotes the discrete (ground truth) value associated with the ith feature. We are particularly interested in the setting where several features are correlated with the label y at varying strengths.

3.1 Evaluating feature quality

In this work, we seek to compare the *feature quality* of the representations learned by different methods. To measure feature quality, our idea is to train a linear probe on the representation to discriminate the corresponding feature, and measure its performance (Rosenfeld et al., 2022; Kirichenko et al., 2022). Importantly, the training must be done on a dataset only the desired feature is correlated with the label, so that probe only picks up only this feature from the representation.

Formally, we construct the linear probe dataset as follows. Consider a *balanced* distribution D_{bal} where there is an equal number of points from each configuration of the features, i.e., there are $|\mathcal{A}_1| \times |\mathcal{A}_2| \times \ldots |\mathcal{A}_m|$ subpopulations that have equal probability masses in D_{bal}. Then, for any feature i, we define the *feature-probing error* for this representation in terms the error of a linear probe u in predicting the true feature value $a_i(x)$ when trained on D_{bal}. Formally,

$$\text{ProErr}_i(\theta) \stackrel{\text{def}}{=} \mathbb{E}_{x \sim D_{\text{bal}}}[\ell_{\text{0-1}}(u^\top \phi_\theta(x), a_i(x))], \text{ for } u = \arg\min \hat{E}_{x \sim D_{\text{bal}}}[\ell(u^\top a_i(x))], \quad (1)$$

where $\hat{E}_{x \sim D_{\text{bal}}}$ corresponds to the empirical distribution over training data from D_{bal} used to train the linear probe and ℓ is some suitable classification loss such as the logistic loss.

3.2 Training algorithms

Empirical risk minimization via Stochastic Gradient Descent (SGD). Stochastic gradient descent (SGD) is the *de facto* approach to minimizing the empirical risk over training data. Given a batch of training samples, $\{(x_1, y_1), \ldots (x_B, y_B)\}$, a loss function ℓ, and model $w = (V, \theta)$, we define the empirical batch loss $\hat{\mathcal{L}}(w) \stackrel{\text{def}}{=} (1/B) \sum_{i=1}^{B} \ell(f(w; x_i), y_i)$. The SGD update is:

$$w \leftarrow w - \eta \nabla_w \hat{\mathcal{L}}(w), \quad (2)$$

Table 1: Comparing the in-distribution testing error and balanced distribution probing error for each feature for SAM and SGD, as defined in Section 3. For CelebA, the hard feature is gender, for Waterbirds, it is background, for CIFAR-MNIST, it is CIFAR, and for FMNIST-MNIST, it is FMNIST.

	Test	Easy ft. probe	Hard ft. probe	Test	Easy ft. probe	Hard ft. probe
	CelebA			Waterbirds		
SGD	4.7 ± 0.07	10.9 ± 1.15	20.9 ± 0.93	5.2 ± 0.13	11.7 ± 2.28	21.8 ± 0.99
SAM	$\mathbf{4.3 \pm 0.10}$	$\mathbf{8.7 \pm 0.51}$	$\mathbf{15.1 \pm 1.10}$	$\mathbf{4.6 \pm 0.12}$	$\mathbf{7.8 \pm 0.50}$	$\mathbf{19.7 \pm 2.00}$
	CIFAR-MNIST			FMNIST-MNIST		
SGD	0.1 ± 0.02	0.1 ± 0.02	12.7 ± 1.05	0.5 ± 0.52	5.8 ± 4.15	11.6 ± 1.93
SAM	0.0 ± 0.01	0.1 ± 0.03	$\mathbf{10.2 \pm 0.51}$	$\mathbf{0.0 \pm 0.02}$	$\mathbf{0.3 \pm 0.10}$	$\mathbf{10.5 \pm 0.36}$

where $\eta > 0$ is the learning rate. Unless otherwise mentioned, we use the cross-entropy loss for multi-class classification and logistic loss for binary classification.

Sharpness-Aware Minimization (SAM). In recent years, SAM has been proposed as an alternative to SGD. For model weights w, the SAM update is:

$$w \leftarrow w - \eta \sum_{i=1}^{B} \nabla_w \hat{\mathcal{L}}(w)\big|_{\tilde{w}}, \text{ where } \tilde{w} \stackrel{\text{def}}{=} w + \rho \nabla_w \hat{\mathcal{L}}(w) / \|\nabla_w \hat{\mathcal{L}}(w)\|_2, \tag{3}$$

for some perturbation radius $\rho > 0$. We refer the reader to Foret et al. (2020) for a derivation.

The SAM phantom parameter. Comparing SGD (Equation 2) and SAM (Equation 3), notice that SAM takes a descent step in the direction of the gradient computed at different point: $\tilde{w}$, which we call the *phantom parameter*. The phantom parameter is computed by taking a ascent step of fixed norm in the direction of the gradient at the current parameter w. This is an important characterization which we exploit in the rest of this paper. By studying how the phantom parameter $\tilde{w}$ relates to the real parameter w, we can study how SAM changes the learning trajectory directly.

4 SAM IMPROVES FEATURE DIVERSITY

In this section, we demonstrate that SAM can empirically improve the quality of multiple redundant features, even when SGD would fail as a result of the simplicity bias (Shah et al., 2020). This quality is particularly important when the features relevant to downstream performance are unknown at training time, and thus necessary to learn a high quality representation of all available features. We evaluate SAM on datasets in which multiple features have been labeled, where SGD tends to learn a higher quality representation of the easier feature. With multiple labeled features, we can evaluate the quality of the representation of each feature individually. We first describe our experimental setup, and then present our results in which we compare the feature quality of SAM and SGD.

4.1 DATASETS AND MODELS

Datasets. We use four datasets in our experiments each annotated by two features: CelebA (Liu et al., 2015), Waterbirds (Sagawa et al., 2019), CIFAR-MNIST (binary) (Shah et al., 2020), and FMNIST-MNIST (5-class) (Kirichenko et al., 2022). The simplicity bias of SGD suggests that the easier of the two features will be learned better. Thus, we refer to the feature which attains the lower probing error for SGD trained models as the "easy" feature and the other has the "hard" feature. We describe the datasets in detail in Appendix B.

Training setup. Following prior work (Kirichenko et al., 2022; Sagawa et al., 2019), we train an ImageNet-pretrained ResNet-18 on CelebA and Waterbirds that has been initialized with weights pretrained on ImageNet. For CIFAR-MNIST, we train a three-layer MLP from scratch. We apply standard data augmentation and weight decay (detailed in Appendix B), and we tune both algorithms over different learning rates, and for SAM, we tune over the ρ parameter. We select the optimal hyperparameters based on the validation set. For CIFAR-MNIST and CelebA, we train for 20 epochs, and for Waterbirds we train for 100 epochs. We repeat each experiment four times using different random seeds, and report the mean and standard deviation of the errors.

4.2 Comparing SAM and SGD

Our goal is to quantify the quality of the features learned by SGD-trained models vs. SAM-trained models.To this end, in all our datasets, we evaluate both the easy and hard features each model in terms of their feature probing error. As described in Section 3, we compute the probing error by training a last-layer probe on a small dataset in which the feature of interest is uncorrelated with the other feature. Then, we report the accuracy of this probe on a holdout test dataset in which the feature of interest is again uncorrelated with the other feature. In accordance with our main hypothesis, we find that SAM consistently achieves lower probing error for both features on all our datasets, in comparison to SGD (Table 1). We have thus verified that SAM implicitly improves the quality of multiple redundant features.

This suggests that even though SAM tries to optimize for the loss (and its sharpness) on the training distribution—and it does so successfully—under the hood, it also somehow improves feature quality on multiple *different*, balanced distributions in which the correlation between features is broken and examples are labeled by a feature of interest. We are interested in understanding how this unexpected "under the hood" improvement of the feature quality arises in SAM.

5 Understanding SAM feature learning in a toy setup

In this section, we design a *minimal*, controlled setup that helps us isolate two core mechanisms within SAM. In our setup, data has an easier and a harder feature that are both predictive of the label. SGD exhibits the simplicity bias and learns a poor quality representation of the harder feature. In contrast, we show that SAM learns a good representation of both features.

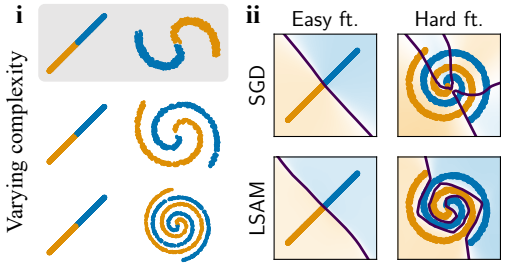

Figure 1: Illustration of the toy example. (**i**) The toy data distribution. We vary the complexity of the spiral component of the data by tightening the spiral. (**ii**) Decision boundary of classifiers trained with SGD and with LSAM along a 2D slice where the other feature.

5.1 A minimal setup

Feature distribution. We consider a 4D setup $\mathcal{X} = \mathbb{R}^4$ where the first two coordinates correspond to a *linear feature* (the "easier" feature) and the second two coordinates correspond to a *spiral feature* (the "harder" feature). We denote the individual components $x = [x_{\text{easy}}, x_{\text{hard}}]$. The associated attributes (see Section 3) for each feature $a_1(x), a_2(x)$ are such that $a_1(x) = 1$ if x resides on the right branch of the linear feature, and zero otherwise; similarly, $a_2(x) = 1$ if x resides on the right spiral, and zero otherwise. We assume a binary label $y \in \{-1, 1\}$ such that each feature independently is fully predictive of it i.e., $y = a_1(x)$ and $y = a_2(x)$. We can vary the *complexity* of the spiral feature by varying the number of rotations of the spiral. Here, the complexity refers to the amount of rotation of the spiral, in degrees.

Disentangled architecture. We are interested in a setting where it is possible to *precisely* measure the quality of representations. To this end, we create a network with an explicitly disentangled representation space (rather than hoping that a trained linear probe would precisely disentangle it). Specifically, we define an architecture whose last-layer representation takes the form

$$\Phi_\theta(x) \stackrel{\text{def}}{=} [\Phi_{\text{easy}}(x), \Phi_{\text{hard}}(x)] \stackrel{\text{def}}{=} [\Phi_\theta([x_{\text{easy}}, \mathbf{0}]), \Phi_\theta([\mathbf{0}, x_{\text{hard}}])], \tag{4}$$

where $\Phi_\theta \colon \mathbb{R}^4$ to $\mathbb{R}$ is a three-layer neural network. This is followed by last layer weights $v = [v_{\text{easy}}, v_{\text{hard}}]$ such that the final classification takes the form, $\text{sign}(f_w(x)) = \text{sign}(v_{\text{easy}}\Phi_{\text{easy}}(x) + v_{\text{hard}}\Phi_{\text{hard}}(x))$. where $w = (v, \theta)$ are the parameters of the neural network. Then, we can think of our feature probe for either features as simply Φ_{easy} and Φ_{hard} respectively.

Probing error. Since we have a disentangled representation space, measuring the probing error is easy—there is no need to train a linear probe, since each feature is represented separately in the last layer. We can simply measure the error $\ell_{0\text{-}1}(\Phi_{\text{easy}}(x), y)$ or $\ell_{0\text{-}1}(\Phi_{\text{hard}}(x), y)$ to compute the probing error for the easy and hard feature, respectively.

LSAM: A simplification of SAM. Recall that SAM involves taking a phantom gradient ascent step to reach parameters, based off of which the actual descent step is calculated. While the SAM phantom step perturbs all parameters of the networks, for this controlled study, we consider a minimal version of SAM where we only perturb the last layer V; we call this LSAM. Note though that the final descent of LSAM still updates all parameters; it is only the *phantom* ascent that is restricted to the last layer. Specifically, following the Section 3 notation, the phantom parameters take the following simple form:

$$\tilde{v} \stackrel{\text{def}}{=} v + \rho \nabla_v \hat{\mathcal{L}}(v, \theta) / \|\nabla_v \hat{\mathcal{L}}(v, \theta)\|_2, \quad \tilde{\theta} \stackrel{\text{def}}{=} \theta. \tag{5}$$

This is nearly identical to Equation 3, except that only v is perturbed, and not θ. As we will shortly see (Section 6), the above simplified version of SAM is sufficient (and in fact, helpful) in cleanly identifying our two fundamental mechanisms that can be linked to improved feature quality.

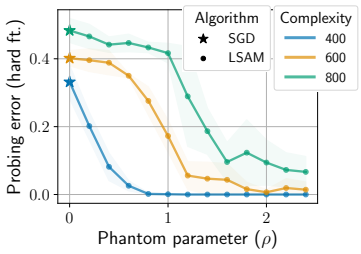

Figure 2: Test-set probing error of harder feature as a function of the phantom parameter ρ, for multiple complexities for the hard feature. SGD corresponds with $\rho = 0$.

5.2 LSAM learns the harder spiral feature, while SGD does not

We note that in this dataset, both SAM and SGD learn the label y perfectly. However, the behaviors are starkly different at the feature-level. In Figure 1, we plot the contour of Φ_{easy} and Φ_{hard} which correspond to how the models have learned either features. We observe, visually, that SGD has no trouble learning the linear feature but has a poor representation of the hard spiral feature. However, LSAM manages to learn a good representation of both features. This implies that despite the fact that SGD and LSAM have the same in-distribution performance, LSAM has a better feature representation of the harder feature.

To quantify this disparity between SAM and SGD, we measure the spiral feature's probing error for varying complexities of the spiral feature (i.e., varying tightness levels), and for varying values of the SAM perturbation radius ρ. We plot the probing error vs. ρ in Figure 2. Note that $\rho = 0$, corresponds to SGD. We find that while the learned spiral feature's quality is poorest when $\rho = 0$ (i.e., SGD), it progressively improves as we increase ρ. In effect, this toy example isolates a core property of SAM that sets it apart from SGD: SAM learns higher quality representations of hard-to-learn features compared to SGD, even in the presence of a fully predictive easy feature. This is analogous to the feature-diversifying effect we observe in real-world datasets in Section 4.

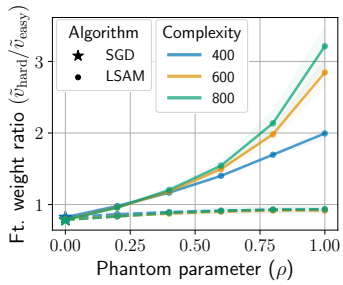

Figure 3: Phantom weight ratio as a function of ρ. We plot $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}}$ (solid lines) and $v_{\text{hard}}/v_{\text{easy}}$ (dashed lines) for different values of ρ when running LSAM. Note that SGD corresponds to when $\rho = 0$. As ρ increases, the ratio $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}}$ increases.

6 Why does SAM learn the hard-to-learn feature?

We have so far established that LSAM achieves significantly lower probing error on a hard-to-learn spiral feature in comparison to SGD (See Figure 2). In this section, we will describe the two mechanisms by which LSAM offers this feature-diversifying benefit without being explicitly trained to do so—we will argue that LSAM re-weights the training examples to increase weight on points that may be helpful for learning the hard feature, and that LSAM increases the effective step size of updates that fit the hard feature. However, before we proceed, we must characterize how LSAM changes the relative weighting of the easy and hard features in the gradient update step. The mechanisms of LSAM that lead to improved feature quality arise as a result of how LSAM changes this relative weighting.

6.1 SAM re-balances phantom parameters

The first step in understanding the mechanisms of LSAM is to understand how the phantom parameters $\tilde{v}_{\text{easy}}$ and $\tilde{v}_{\text{hard}}$ behave as we vary the perturbation radius ρ. We plot the ratio $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}}$ as

a function of ρ in Figure 3. When this ratio is large, the hard feature is upweighted relative to the easy feature. When $\rho = 0$, corresponding with SGD, the ratio is close to 1, indicating that the easy and hard features are weighted equally. However, as we increase ρ, the ratio increases, placing more weight on the hard feature component of the model. This demonstrates that LSAM re-balances the relative weight of the two features in the gradient update step by placing more weight on the hard feature. It is important to note that this re-balancing effect is not the result of the real weight v_{hard} and v_{easy} changing, but rather the result of the perturbation applied by LSAM when computing the phantom parameters (refer to Equation 3). The ratio of the real parameters $v_{\text{hard}}/v_{\text{easy}}$ remains nearly constant as a function of the perturbation radius ρ (Figure 3, dashed line).

6.2 The feature-diversifying effects of SAM

With knowledge that LSAM reweights the ratio $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}}$, our main idea is to decompose the gradient update into two terms in which we can interpret the effect of this reweighting. For binary classification, logistic loss on sample (x_i, y_i), we can write the gradient update to parameters θ as:

$$\nabla_\theta \hat{\mathcal{L}} = \underbrace{\sigma\big(-y_i(v_{\text{easy}}\Phi_{\text{easy}}(x_i) + v_{\text{hard}}\Phi_{\text{hard}}(x_i))\big)}_{\text{Importance weighting } \lambda_i} \cdot \underbrace{\big(v_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i)\big)}_{\text{Sum of feature gradients } g_i}, \tag{6}$$

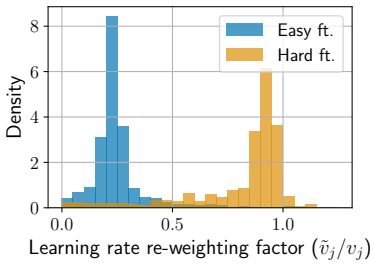

Figure 4: Effective feature learning rate effect plotted as a distribution over batches in the dataset. (**i**) The learning rate re-weighting factor for the easy feature $\tilde{v}_{\text{easy}}/v_{\text{easy}}$ and the hard feature $\tilde{v}_{\text{hard}}/v_{\text{hard}}$, from Equation 6. (**ii**) The hard feature proportional learning rate for the real parameter $v_{\text{hard}}/(v_{\text{easy}} + v_{\text{hard}})$ and the phantom parameter $\tilde{v}_{\text{hard}}/(\tilde{v}_{\text{easy}} + \tilde{v}_{\text{hard}})$.

where σ is the sigmoid function. If we replace v_{hard} and v_{easy} with the phantom parameters $\tilde{v}_{\text{hard}}, \tilde{v}_{\text{easy}}$, we get analogous terms $\tilde{\lambda}_i$ and $\tilde{g}_i$ for LSAM. The scalar *importance weighting* term λ_i represents the contribution of each individual example to the final loss gradient. The vector *sum of feature gradients* term g_i is a weighted combination of the individual feature gradients. The effect of LSAM on the importance weighting term is example-specific: the degree to which each point is upweighted by LSAM $\tilde{\lambda}_i/\lambda_i$ depends on the example. However, the effect of LSAM on the sum of feature gradients term is constant across all examples in a batch: the degree to which the effective learning rate of each feature is up-weighted by LSAM, $\tilde{v}_{\text{hard}}/v_{\text{hard}}$ and $\tilde{v}_{\text{easy}}/v_{\text{easy}}$, is the same for all examples within a batch.

In the remainder of the section, we explore two effects that arise when $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}} \gg v_{\text{hard}}/v_{\text{easy}}$ that enable LSAM to learn the harder feature.

Effect one: the importance weighting effect. We consider the effect of increasing $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}}$ on the $\tilde{\lambda}_i$ term in Equation 6, which corresponds to how heavily each point is weighted in the gradient. At a high level, we argue that this effect leads to two key beneficial behaviors that can be tested experimentally. First, examples are weighted more uniformly. Second, examples that are well fit by the easy feature but poorly fit by the hard feature are upweighted. Both of these effects can promote the learning of the harder feature by encouraging points additional points to contribute to the learning of the harder feature that would otherwise have contributed negligibly. In particular, the second effect leads to the up-weighting of points that are incorrectly fit by the harder feature, but have low loss because they are well fit by the easy feature. Without the up-weighting effect, the effect of these low-loss points on the gradient would be too small to promote the learning of the harder feature (an effect commonly called "gradient starvation"). The effect of LSAM is similar to the intervention imposed by algorithms that explicitly up-weight examples that belong to the minority class in imbalanced datasets (Liu et al., 2021; Sagawa et al., 2019), but it is remarkable that LSAM induces this importance weighting effect *implicitly* without being designed to do so.

In order to confirm the first behavior—that points become more uniformly weighted—we visualize the distribution of the importance weights as a Lorenz curve, which plots the cumulative fraction of the total importance contributed by the top k points, as a function of k. When the importance weights are completely uniform, the Lorenz curve will be a diagonal line. We plot this hypothetical Lorenz curve as a dotted line to compare to the actual Lorenz curves: the closer each curve is to this dotted line, the more uniform the importance weights are across all points. We aim to make two

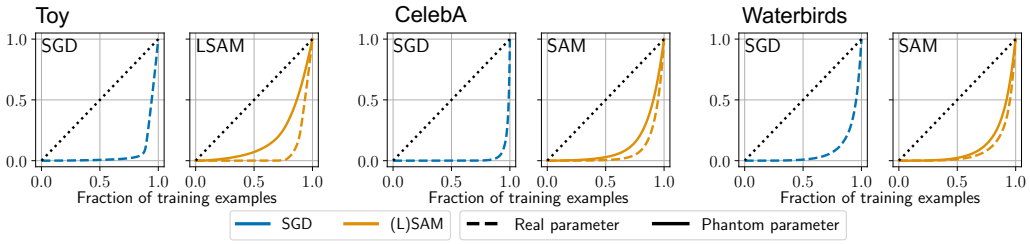

Figure 5: Lorenz curves for the real and phantom importance weight λ_i and $\tilde{\lambda}_i$. The dotted diagonal line represents the Lorenz curve for a uniform distribution. The closer this curve is to this diagonal, the more equally the importance weights are spread. In blue, we plot the Lorenz curves for an SGD checkpoint. In orange, we plot the Lorenz curves for an LSAM checkpoint. The update step gradient is computed at real parameter for SGD, and the phantom parameter for SAM. We include the curves for the toy (left), CelebA (center), and Waterbirds (right). We observe that within the SAM checkpoints, the weights of points evaluated at the phantom parameter are closer to uniform than when evaluate at the real parameter. Further, in comparison to an SGD checkpoint, the phantom parameter of SAM weights points more uniformly.

comparisons. First, we want to compare the behavior of an LSAM step to an SGD given fixed model parameters. Recall that the gradient computed at the real parameter corresponds with an SGD update, and the gradient computed at the phantom parameter corresponds with an LSAM update. Thus, we compare the Lorenz curves of the importance weights computed using the real parameters and computed using the phantom phantom parameters for a model that has been (incompletely) trained (Figure 5, left, comparing dashed to solid). The phantom parameter has more uniform importance weights than the real parameter. However, even though an LSAM step has a more balanced update than an SGD step with fixed weights, it is possible that a model that is trained with SGD might still lead to more balanced updates than a model that is trained with LSAM by learning different weights. Thus, we compare the Lorenz curves of the update step for a model that has been trained with SGD and one that has been trained with LSAM (Figure 5, left, comparing solid line in LSAM plot to dashed line in SGD plot). Again, the importance weights of the phantom parameter are more uniform than the importance weights of the real parameter for SGD. This verifies our claim that LSAM induces the importance weighting effect in the toy setting and that the importance weights are more uniformly weighted under LSAM in both settings. We plot additional Lorenz curves corresponding to different points during training in Appendix G.

Next, we confirm the second behavior: that points that are well fit by the easy feature but poorly fit by the hard feature are upweighted. We need a metric to measure the the degree to which the easy and hard features are fit by the model. In our toy setting, we can compute the contribution of each feature explicitly. We can thus measure the degree to which the easy and hard features are fit by the model as their contribution signed by the label: $y\Phi_{\text{easy}}(x)$ and $y\Phi_{\text{hard}}(x)$. When either term is positive, the point is classified correctly by the corresponding feature, with the magnitude of the term indicating the margin. In order to summarize how points are weighted as a function of the signed contributions, we partition the points into bins based on $y\Phi_{\text{easy}}(x)$ and $y\Phi_{\text{hard}}(x)$. Thus, we plot the median importance weight for each bin, which gives us the relationship between the importance weight of each example and their signed feature contribution. We see that typical examples that are poorly-fit by the easy feature ($y\Phi_{\text{easy}}(x)$ is small) are assigned less importance by SAM, and examples that are poorly-fit by the hard feature ($y\Phi_{\text{hard}}(x)$ is small) are assigned more importance by SAM (Figure 6). This verifies our claim that LSAM up-weights points that are well-fit by the easy feature but poorly-fit by the hard feature in the toy setting. We plot additional importance weight plots corresponding to different points during training in Appendix G.

Verifying the importance weighting effect in a real-world setup. We have thus far shown that LSAM induces the importance weighting effect in the toy setting. However, what happens when we relax our assumptions and move to a more realistic setting? We aim to test the importance weighting effect when we run the usual SAM algorithm without a disentangled representation space on real-world datasets. We can evaluate importance weight uniformity directly. We plot the Lorenz curve for the importance weights evaluated at both the real parameter and the phantom parameter for CelebA and Waterbirds in Figure 5. We see that the points are more uniformly weighted under

a SAM perturbation, suggesting that the importance weighting effect is present in the real-world datasets as well.

To verify that LSAM up-weights points that are well-fit by the easy feature but poorly-fit by the hard feature, we used the disentangled representation space to measure the degree to which the easy and hard features are fit by the model. However, in the real setting, we cannot directly measure the contribution of each feature. Instead, we rely on extracting the feature contributions from the linear probes v_{easy} and v_{hard} that best classifies the respective feature (refer to Section 3). With these probes, we can define the signed contribution of the easy and hard feature as $yv_{\text{easy}}^\top \Phi(x)$ and $yv_{\text{hard}}^\top \Phi(x)$, respectively. Thus, we plot the median importance weight for both features as a function of $yv_{\text{easy}}^\top \Phi(x)$ and $yv_{\text{hard}}^\top \Phi(x)$ for CelebA and Waterbirds. Analogous to the toy experiment, we see that examples that are poorly-fit by the easy feature are assigned less importance by SAM, and examples that are poorly-fit by the hard feature are assigned more importance by SAM (Figure 6).

Effect two: the learning rate scaling effect. We now turn our attention to the second term, $\tilde{g}_i$, from Equation 6. The $\tilde{g}_i$ term is a weighted combination of the feature gradients with weights $\tilde{v}_{\text{easy}}$ and $\tilde{v}_{\text{hard}}$ corresponding to the easy and hard feature respectively. These weights can be viewed as the "learning rates" as we take gradients steps in the direction of learning different features. Notably, these learning rates are constant across points within a batch. When LSAM causes $\tilde{v}_{\text{hard}}/\tilde{v}_{\text{easy}} \gg v_{\text{hard}}/v_{\text{easy}}$, $\tilde{g}_i$ corresponds with a larger effective step size associated with the hard feature in comparison to g_i. As a result the effective learning rate for the hard feature is higher and leads to more progress on the hard feature. We plot the ratio between the learning rates of the phantom parameters and the real parameters $\tilde{v}_{\text{hard}}/v_{\text{hard}}$ and $\tilde{v}_{\text{easy}}/v_{\text{easy}}$ for each feature in Figure 4. We see that LSAM increases the learning rate for hard features and decreases the learning rate for each feature. We call this the *learning rate scaling effect*. Unlike the importance weighting effect which acts on each data point separately, the learning rate scaling effect is more global across the entire training batch.

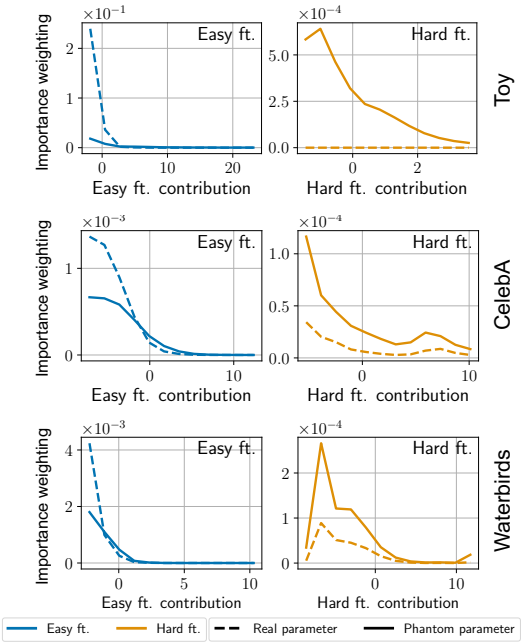

Figure 6: Median importance weighting as a function of the contribution of each feature. We partition the data into bins based on the contribution of the easy and hard features ($yv_{\text{easy}}\Phi_{\text{easy}}$ and $yv_{\text{easy}}\Phi_{\text{hard}}$), as defined in Section 6.2. For each of these bins, we plot the median importance weight term λ_i for the points in the bin. We include the corresponding plots for the toy (top), CelebA (center), and Waterbirds (bottom).

The learning rate effect is difficult to verify in practice. Unfortunately, in the absence of explicitly disentangled architecture (like in our toy setting), it is difficult to estimate the learning rate scaling effect. The key challenge in achieving a similar observation of the learning-rate effect on real data is understanding precisely how features are represented. Since, SAM perturbs the weights at every layer (unlike LSAM), we would need to understand how each feature is represented at every layer of the neural network. Based on the general principle that SAM perturbs weights as a function of how much they affect the output, we suspect, intuitively, that well-learned features will be inhibited by SAM at every layer. While we cannot verify the effect explicitly for a realistic setup, we believe it contributes to SAM's gains in feature probing error.

Determining the contribution of each effect. To recap, we show that LSAM induces (i) an importance weighting effect and (ii) a learning rate scaling effect, both of which in turn arise from by re-balancing the phantom parameters. We have verified that each of these effects are present in the toy example, and we would intuitively expect each effect to have a feature-diversifying effect. But

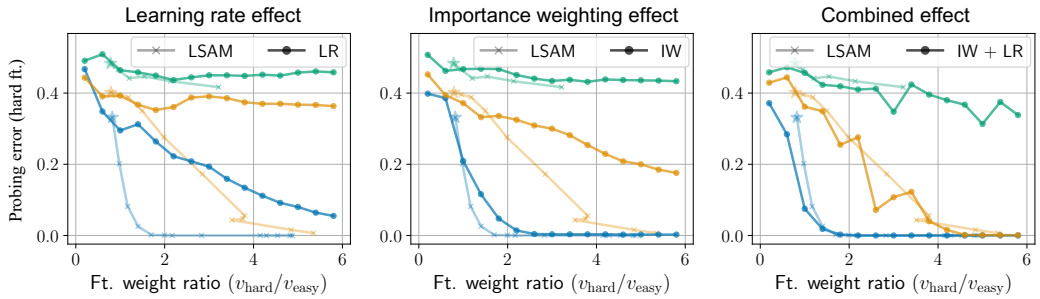

Figure 7: Simulations of the learning rate, importance weighting, and combined effects. Both effects offer gains in hard feature probing accuracy and together account for all the benefits offered by LSAM in the toy setting. See Appendix C for a detailed description of the simulations.

how do these effects compare in improvement in feature quality, and which effect if any is more important? Furthermore, are these effects simply correlations caused by SAM's improved features, or do these effects causally improve SAM?

We answer these questions in the toy setting, where it is possible to explicitly add or remove each effect. To isolate each effect, we train with an algorithm that applies only one effect at a time: either the importance weighting effect or the learning rate effect.

To isolate the importance weighting effect, we train with an algorithm in which we fix the importance weighting term to a constant value,

$$\lambda_i \coloneqq \sigma\big(-y_i(v^*_{\text{easy}}\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\Phi_{\text{hard}}(x_i)\big) \qquad g_i \text{ is as usual for SGD} \tag{7}$$

where $v^*_{\text{easy}} = 1$ and v^*_{hard} is a hyperparameter.

To isolate the learning rate effect, we fix the sum of feature gradients term,

$$g_i = v^*_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i) \qquad \lambda_i \text{ is as usual for SGD,} \tag{8}$$

again, where $v^*_{\text{easy}} = 1$ and v^*_{hard} is a hyperparameter. Finally, we can combine both manual interventions and set

$$\lambda_i \coloneqq \sigma\big(-y_i(v^*_{\text{easy}}\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\Phi_{\text{hard}}(x_i)\big) \quad g_i \coloneqq v^*_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i). \tag{9}$$

We run SGD to train the model, but with the modified coefficients (i.e., Equation 6, but with λ_i and g_i replaced as described). The algorithms are described more precisely in Appendix C.

In order to compare these algorithms with LSAM, we compute the probing error for the hard feature against the ratio between the hard feature contribution and the easy feature contribution. For our algorithms, we have manually set this value $v^*_{\text{hard}}/v^*_{\text{easy}}$ (see Equations 7–9). For LSAM, we consider this value to be the mean value of $v_{\text{hard}}/v_{\text{easy}}$ over training (From Figure 3). This leaves both our new algorithms and LSAM to be directly comparable as a function of this ratio, which we plot (Figure 7). We see that both individual interventions (Figure 7, left and center) improve the probing error for the hard feature. This suggests that both effects are causal and also significant (offer comparable gains) in explaining LSAM's improved spiral probing error in the toy setting. Further, the combined effect (Figure 7, right) matches the performance of LSAM and thus appears to provide a complete picture of understanding LSAM in our toy.

7 Discussion

In this work, offer an interpretable understanding of the dynamics of SAM that lead to the improvement of feature diversity, in contrast with the usual sharpness-based arguments. We have shown how SAM can promote balanced feature learning in the presence of multiple redundant features, and that this can lead to improved performance on out-of-distribution tasks. We have also demonstrated that SAM leads to improvements in feature quality for transfer learning on real data, including the Waterbirds dataset, CelebA, CIFAR-MNIST, and DomainBed. We hope that our insights provide a new perspective on the dynamics of SAM without relying on the flatness-based arguments, and that this work will inspire future work to further understand the mechanisms of SAM and will foster new algorithms for improving downstream performance on out-of-distribution data in similar contexts.

ACKNOWLEDGEMENTS

We would like to thank Christina Baek, Jeremy Cohen, and Suhas Kotha for their feedback.

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant No. DGE2140739. Any opinion, findings, and conclusions or recommendations expressed in this material are those of the authors(s) and do not necessarily reflect the views of the National Science Foundation. This work was supported in part by the AI2050 program at Schmidt Sciences (Grant #G2264481). We gratefully acknowledge the support of Apple.

REFERENCES

Momin Abbas, Quan Xiao, Lisha Chen, Pin-Yu Chen, and Tianyi Chen. Sharp-maml: Sharpness-aware model-agnostic meta learning. In *International Conference on Machine Learning*, pp. 10–32. PMLR, 2022.

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095*, 2021.

Michael A Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4845–4854, 2019.

Maksym Andriushchenko and Nicolas Flammarion. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pp. 639–668. PMLR, 2022.

Maksym Andriushchenko, Dara Bahri, Hossein Mobahi, and Nicolas Flammarion. Sharpness-aware minimization leads to low-rank features. *arXiv preprint arXiv:2305.16292*, 2023a.

Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. *arXiv preprint arXiv:2302.07011*, 2023b.

Saeid Asgari, Aliasghar Khani, Fereshte Khani, Ali Gholami, Linh Tran, Ali Mahdavi Amiri, and Ghassan Hamarneh. Masktune: Mitigating spurious correlations by forcing to explore. *Advances in Neural Information Processing Systems*, 35:23284–23296, 2022.

Dara Bahri, Hossein Mobahi, and Yi Tay. Sharpness-aware minimization improves language model generalization. *arXiv preprint arXiv:2110.08529*, 2021.

Kayhan Behdin and Rahul Mazumder. Sharpness-aware minimization: An implicit regularization perspective. *arXiv preprint arXiv:2302.11836*, 2023.

Wieland Brendel and Matthias Bethge. Approximating cnns with bag-of-local-features models works surprisingly well on imagenet. *arXiv preprint arXiv:1904.00760*, 2019.

Junbum Cha, Sanghyuk Chun, Kyungjae Lee, Han-Cheol Cho, Seunghyun Park, Yunsung Lee, and Sungrae Park. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Jiawei Du, Hanshu Yan, Jiashi Feng, Joey Tianyi Zhou, Liangli Zhen, Rick Siow Mong Goh, and Vincent YF Tan. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021.

Jiawei Du, Daquan Zhou, Jiashi Feng, Vincent Tan, and Joey Tianyi Zhou. Sharpness-aware training for free. *Advances in Neural Information Processing Systems*, 35:23439–23451, 2022.

Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

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

Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

Suchin Gururangan, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel R Bowman, and Noah A Smith. Annotation artifacts in natural language inference data. *arXiv preprint arXiv:1803.02324*, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Katherine Hermann and Andrew Lampinen. What shapes feature representations? exploring datasets, architectures, and training. *Advances in Neural Information Processing Systems*, 33: 9995–10006, 2020.

Katherine Hermann, Ting Chen, and Simon Kornblith. The origins and prevalence of texture bias in convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:19000–19015, 2020.

Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

Pavel Izmailov, Polina Kirichenko, Nate Gruver, and Andrew G Wilson. On feature learning in the presence of spurious correlations. *Advances in Neural Information Processing Systems*, 35: 38516–38532, 2022.

Samyak Jain, Sravanti Addepalli, Pawan Sahu, Priyam Dey, and R Venkatesh Babu. Dart: Diversify-aggregate-repeat training improves generalization of neural networks. *arXiv preprint arXiv:2302.14685*, 2023.

Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

Jean Kaddour, Linqing Liu, Ricardo Silva, and Matt J Kusner. When do flat minima optimizers work? *Advances in Neural Information Processing Systems*, 35:16577–16595, 2022.

Dimitris Kalimeris, Gal Kaplun, Preetum Nakkiran, Benjamin Edelman, Tristan Yang, Boaz Barak, and Haofeng Zhang. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.

Simran Kaur, Jeremy Cohen, and Zachary Chase Lipton. On the maximum hessian eigenvalue and generalization. In *Proceedings on*, pp. 51–65. PMLR, 2023.

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

Polina Kirichenko, Pavel Izmailov, and Andrew Gordon Wilson. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.

Alexander Kolesnikov and Christoph H Lampert. Improving weakly-supervised object localization by micro-annotation. *arXiv preprint arXiv:1605.05538*, 2016.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16*, pp. 491–507. Springer, 2020.

Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671, 2019.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pp. 5905–5914. PMLR, 2021.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Evan Z Liu, Behzad Haghgoo, Annie S Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning*, pp. 6781–6792. PMLR, 2021.

Yong Liu, Siqi Mai, Xiangning Chen, Cho-Jui Hsieh, and Yang You. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12360–12370, 2022.

Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 181–196, 2018.

R Thomas McCoy, Ellie Pavlick, and Tal Linzen. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. *arXiv preprint arXiv:1902.01007*, 2019.

Xuran Meng, Yuan Cao, and Difan Zou. Per-example gradient regularization improves learning signals from noisy data. *arXiv preprint arXiv:2303.17940*, 2023.

Mazda Moayeri, Phillip Pope, Yogesh Balaji, and Soheil Feizi. A comprehensive study of image classification model sensitivity to foregrounds, backgrounds, and visual attributes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 19087–19097, 2022.

Depen Morwani, Jatin Batra, Prateek Jain, and Praneeth Netrapalli. Simplicity bias in 1-hidden layer neural networks. *arXiv preprint arXiv:2302.00457*, 2023.

Clara Na, Sanket Vaibhav Mehta, and Emma Strubell. Train flat, then compress: Sharpness-aware minimization learns more compressible models. *arXiv preprint arXiv:2205.12694*, 2022.

Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? *Advances in neural information processing systems*, 33:512–523, 2020.

Matteo Pagliardini, Martin Jaggi, François Fleuret, and Sai Praneeth Karimireddy. Diversity through disagreement for better transferability. In *NeurIPS 2022 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2022.

Sinno Jialin Pan and Qiang Yang. A survey on transfer learning ieee transactions on knowledge and data engineering. *22 (10)*, 1345, 2009.

Mohammad Pezeshki, Oumar Kaba, Yoshua Bengio, Aaron C Courville, Doina Precup, and Guillaume Lajoie. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.

Gregory Plumb, Marco Tulio Ribeiro, and Ameet Talwalkar. Finding and fixing spurious patterns with explanations. *arXiv preprint arXiv:2106.02112*, 2021.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pp. 5301–5310. PMLR, 2019.

Harsh Rangwani, Sumukh K Aithal, Mayank Mishra, et al. Escaping saddle points for effective generalization on class-imbalanced data. *Advances in Neural Information Processing Systems*, 35:22791–22805, 2022.

Amir Rosenfeld, Richard Zemel, and John K Tsotsos. The elephant in the room. *arXiv preprint arXiv:1808.03305*, 2018.

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

Shiori Sagawa, Pang Wei Koh, Tatsunori B Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.

Luca Scimeca, Seong Joon Oh, Sanghyuk Chun, Michael Poli, and Sangdoo Yun. Which shortcut cues will dnns choose? a study from the parameter-space perspective. *arXiv preprint arXiv:2110.03095*, 2021.

Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33:9573–9585, 2020.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 806–813, 2014.

Rakshith Shetty, Bernt Schiele, and Mario Fritz. Not using the car to see the sidewalk–quantifying and controlling the effects of context in classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8218–8226, 2019.

Sahil Singla and Soheil Feizi. Salient imagenet: How to discover spurious features in deep learning? *arXiv preprint arXiv:2110.04301*, 2021.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pp. 843–852, 2017.

Hao Sun, Li Shen, Qihuang Zhong, Liang Ding, Shixiang Chen, Jingwei Sun, Jing Li, Guangzhong Sun, and Dacheng Tao. Adasam: Boosting sharpness-aware minimization with adaptive learning rate and momentum for training deep neural networks. *arXiv preprint arXiv:2303.00565*, 2023.

Damien Teney, Ehsan Abbasnejad, Simon Lucey, and Anton Van den Hengel. Evading the simplicity bias: Training a diverse set of models discovers solutions with superior ood generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16761–16772, 2022.

Dong Wang, Yicheng Liu, Liangji Fang, Fanhua Shang, Yuanyuan Liu, and Hongying Liu. Balanced gradient penalty improves deep long-tailed learning. In *Proceedings of the 30th ACM International Conference on Multimedia*, pp. 5093–5101, 2022.

Pengfei Wang, Zhaoxiang Zhang, Zhen Lei, and Lei Zhang. Sharpness-aware gradient matching for domain generalization. *arXiv preprint arXiv:2303.10353*, 2023.

Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.

Kaiyue Wen, Tengyu Ma, and Zhiyuan Li. Sharpness minimization algorithms do not only minimize sharpness to achieve better generalization. *arXiv preprint arXiv:2307.11007*, 2023.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Kai Xiao, Logan Engstrom, Andrew Ilyas, and Aleksander Madry. Noise or signal: The role of image backgrounds in object recognition. *arXiv preprint arXiv:2006.09994*, 2020.

Yao-Yuan Yang, Chi-Ning Chou, and Kamalika Chaudhuri. Understanding rare spurious correlations in neural networks. *arXiv preprint arXiv:2202.05189*, 2022.

Haotian Ye, James Zou, and Linjun Zhang. Freeze then train: Towards provable representation learning under spurious correlations and feature noise. In *International Conference on Artificial Intelligence and Statistics*, pp. 8968–8990. PMLR, 2023.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Xingxuan Zhang, Renzhe Xu, Han Yu, Hao Zou, and Peng Cui. Gradient norm aware minimization seeks first-order flatness and improves generalization. *arXiv preprint arXiv:2303.03108*, 2023.

Yang Zhao, Hao Zhang, and Xiuyuan Hu. Penalizing gradient norm for efficiently improving generalization in deep learning. In *International Conference on Machine Learning*, pp. 26982–26992. PMLR, 2022.

Juntang Zhuang, Boqing Gong, Liangzhe Yuan, Yin Cui, Hartwig Adam, Nicha Dvornek, Sekhar Tatikonda, James Duncan, and Ting Liu. Surrogate gap minimization improves sharpness-aware training. *arXiv preprint arXiv:2203.08065*, 2022.

A Additional related work

Shortcut learning and spurious correlations. Many works have demonstrated shortcut learning in neural networks image and natural language classification tasks (Geirhos et al., 2018; 2020; Scimeca et al., 2021; Yang et al., 2022; Hermann et al., 2020; Hermann & Lampinen, 2020; Brendel & Bethge, 2019; Alcorn et al., 2019; Shetty et al., 2019; Singla & Feizi, 2021; Rosenfeld et al., 2018; Kolesnikov & Lampert, 2016; Moayeri et al., 2022; Gururangan et al., 2018; McCoy et al., 2019; Xiao et al., 2020; Sagawa et al., 2019). In contrast, we focus purely on feature *probing* error which looks at how well a feature is captured in the representation irrespective of how much the final classifier relies on different features. We also care about probing error with respect to all predictive features and do not delineate specific features as spurious. Recent works (Kirichenko et al., 2022; Izmailov et al., 2022; Rosenfeld et al., 2022) have shown that just last layer retraining on a suited dataset can significantly overcome reliance on spurious correlations suggesting that the non-spurious features are reasonably well-learnt by the representation.

Feature diversity and finetuning. Multiple methods have been proposed to improve feature diversity to evade the simplicity bias (Teney et al., 2022), ignore spurious features (Asgari et al., 2022), improve transfer learning (Pagliardini et al., 2022), and improve generalization (Jain et al., 2023). Fine-tuning and other retraining has been a popular method to adopt learned features to tasks other than the pre-training objective (Pan & Yang, 2009). Retraining has been adopted to recover from failure modes (Rosenfeld et al., 2022; Kirichenko et al., 2022; Kumar et al., 2022) and to improve generalization on novel tasks (Radford et al., 2021; Sharif Razavian et al., 2014; Huh et al., 2016; Sun et al., 2017; Mahajan et al., 2018; Kolesnikov et al., 2020; Zhai et al., 2019; Dosovitskiy et al., 2020; Neyshabur et al., 2020; Abnar et al., 2021; Kornblith et al., 2019).

Alternative SAM variants. Multiple variants have been proposed to generalization notions of flatness (Kwon et al., 2021; Zhuang et al., 2022), to improve the efficiency of SAM (Du et al., 2022; 2021; Liu et al., 2022), and to generalize SAM to different optimizers (Sun et al., 2023).

B Omitted experimental details

Description of datasets. CelebA is a large-scale face attributes dataset with 40 binary attributes. Following Kirichenko et al. (2022), we train a classifier to predict the "hair color" attribute, with values {blond, dark-hair}. However, the "gender" attribute, with values {female, male} is highly correlated with "hair color" due to the imbalance in the dataset, and thus can additionally be useful to predict the label, for the training distribution. Waterbirds is a synthetic dataset consisting of images of birds superimposed onto different backgrounds. The data is annotated by "bird type", with values {water bird, land bird}, which refers to whether the pictured bird primarily lives in water or on land. The "background" attribute, with values {water, land}, is highly correlated with "bird type". CIFAR-MNIST is a dataset consisting images from MNIST (LeCun et al., 1998) and CIFAR-10 (Krizhevsky et al., 2009) that have been concantenated together, where the label of the CIFAR image and the label of the MNIST image are perfectly correlated for all training examples. Following Shah et al. (2020), we restrict the dataset to the binary setting. The "CIFAR" attribute can attain values {airplane, automobile} and refers to the label of the CIFAR component of the image. Similarly, the "MNIST" attribute can attain values $\{0, 1\}$. For all datasets, we use the standard train/validation/test split, and when a validation set is not provided, we use a random 90/10 split of the training set. For computational efficiency, we down-scale the CelebA images to 64×64 pixels, and the Waterbirds images to 96×96 pixels. For FashionMNIST-MNIST dataset (Xiao et al., 2017), we restrict to the first five classes, associated with the digits 0–5 of MNIST.

Reporting test error. Since the validation and testing datasets of Waterbirds and CelebA differ in distribution from the training set, to be consistent with Kirichenko et al. (2022) when reporting the testing error, we weight the testing error of each group by its corresponding frequency in the training dataset.

Architectures For the experiments involving CIFAR-MNIST and variants, we train on a simple convolutional architecture including three convolutional layers, followed by ReLUs, with a final linear decoding layer. The architecture is defined by the following pseudo-PyTorch:

```
torch.nn.Sequential(
    torch.nn.Conv2d(3, 32, kernel_size=5, stride=2, padding=2),
    torch.nn.ReLU(inplace=True),
    torch.nn.Conv2d(32, 64, kernel_size=3, stride=2, padding=1),
    torch.nn.ReLU(inplace=True),
    torch.nn.Conv2d(64, 128, kernel_size=3, stride=2, padding=1),
    torch.nn.ReLU(inplace=True),
    torch.nn.Flatten(),
    torch.nn.Linear(n_features, num_classes)
)
```

For the experiments involving CelebA and Waterbirds, we use an ImageNet-pretrained ResNet-18 as specified by He et al. (2016).

For the toy experiments, the representation neural network Φ_θ had the architecture specified by the following pseudo-PyTorch:

```
torch.nn.Sequential(
    torch.nn.Linear(n_features, 100),
    torch.nn.LayerNorm(100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, 100),
    torch.nn.LayerNorm(100),
    torch.nn.ReLU(),
    torch.nn.Linear(100, n_classes)
)
```

Parameters and sweeps. For the toy experiments, we choose a constant learning rate of 0.01, a batch size of 5, 300 training points, no momentum, and no weight decay.

For the CIFAR-MNIST and FMNIST-MNIST experiments, we sweep over the learning rates $\{0.01, 0.05, 0.1\}$ and the phantom hyperparameter ρ over $\{0.0, 0.01, 0.03, 0.05, 0.07, 0.1, 0.2\}$. We use a batch size of 100, a cosine learning rate schedule, a momentum parameter of 0.9, and no weight decay. We normalize the images by the mean pixel value. Otherwise, we do not use data augmentation.

For the CelebA and Waterbirds experiments, we sweep over the learning rates $\{0.0005, 0.001, 0.005, 0.01\}$ and the ρ parameter $\{0.0, 0.01, 0.02, 0.05, 0.07\}$. We use a batch size of 128, a cosine learning rate schedule, a momentum parameter of 0.9, and a weight decay of 10^{-4}. We use data augmentation described by the following pseudo-PyTorch:

```
WATERBIRDS_TRANSFORMS_AUGMENT = transforms.Compose([
    transforms.RandomResizedCrop(96,
        scale=(0.7, 1.0),
        ratio=(0.75, 4./3.),
        interpolation=InterpolationMode.BILINEAR),
    transforms.RandomHorizontalFlip(),
    transforms.ToTensor(),
    transforms.Normalize(WATERBIRDS_MEAN, WATERBIRDS_STD)
])

CELEBA_TRANSFORM = transforms.Compose([
    transforms.Resize((64, 64)),
    transforms.ToTensor(),
    transforms.Normalize(CELEBA_MEAN, CELEBA_STD)
])
```

C Additional details for the toy setting

Algorithms for the interventions To simulate only the importance weighting effect, we re-weight points in the importance weighting term λ_i to the weight of each point computed as if we had fixed the feature ratio. Similarly, to simulate the learning-rate effect, we re-weight each feature in the sum of feature gradients term g_i by the weight that would assigned if we had fixed the feature ratio. In order to verify that each effect is important and to quantify the relative strenght of each effect, we intervene individually on each effect, separately, by fixing the ratio $v_{\text{easy}}/v_{\text{hard}}$, *but only when computing the term corresponding with that effect*. This means that the loss gradient for the importance weighting intervention is computed as,

$$g_{\text{i.w.}}(\theta) = \sum_{i=1}^{B} \sigma_{y_i} \left(v^*_{\text{easy}}\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\Phi_{\text{hard}}(x_i)\right) \left(v_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i)\right), \tag{10}$$

and the loss gradient for feature gradient effective learning rate intervention is computed as,

$$g_{\text{l.r.}}(\theta) = \sum_{i=1}^{B} \sigma_{y_i} \left(v_{\text{easy}}\Phi_{\text{easy}}(x_i) + v_{\text{hard}}\Phi_{\text{hard}}(x_i)\right) \left(v^*_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i)\right). \tag{11}$$

For both equations, v^*_{easy} and v^*_{hard} are fixed and do not change during training. We train with SGD, but we compute the gradient as described in Equations 10 and 11.

To simulate the effects together, we combine the two interventions by fixing the ratio $v_{\text{easy}}/v_{\text{hard}}$ in both terms. This means that the loss gradient for the combined intervention is computed as,

$$g_{\text{c.}}(\theta) = \sum_{i=1}^{B} \sigma_{y_i} \left(v^*_{\text{easy}}\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\Phi_{\text{hard}}(x_i)\right) \left(v^*_{\text{easy}}\nabla_\theta\Phi_{\text{easy}}(x_i) + v^*_{\text{hard}}\nabla_\theta\Phi_{\text{hard}}(x_i)\right). \tag{12}$$

D THEORETICAL INTUITION AND ANALYSIS

In this section, we present a brief discussion of the theoretical intuition behind our results. In particular, we will aim to understand the importance weighting effect.

Preliminaries. For simplicity, we will aim to understand the dynamics when training with a batch is a single example x. We will assume, as is typical, that we have a neural network,

$$f_x(\theta) = w^\top \phi_\theta(x) \tag{13}$$

parameterized by θ, and where $\phi_\theta(x)$ is the representation of x under the neural network. We assume that the neural network is differentiable and well-approximated by a first-order Taylor expansion. We assume that the loss function $\mathcal{L}(\theta) = \exp(-yf(\theta))$ is exponential, though note that cross-entropy is almost identical to exponential, for correctly classified points. Finally, recall from the main body of the paper that we can separate the loss gradient into two terms,

$$\nabla_\theta \mathcal{L}(\theta) = -y \underbrace{\exp(-yf(\theta))}_{\text{importance weighting term } \lambda} \underbrace{\nabla_\theta f(\theta)}_{\text{feature gradient term } g}. \tag{14}$$

We define the phantom importance weighting parameter $\tilde{\lambda}$ as the importance weighting term evaluated at the phantom parameter $\tilde{\theta}$.

Phantom parameter. Recall that the SAM perturbation with exponential loss is defined as,

$$\tilde{\theta} = \theta + \rho \frac{\nabla_\theta \mathcal{L}(\theta)}{\|\nabla_\theta \mathcal{L}(\theta)\|} \tag{15}$$

$$= \theta + \rho \frac{\nabla_\theta \exp(-yf_x(\theta))}{\|\nabla_\theta \exp(-yf_x(\theta))\|} \tag{16}$$

$$= \theta - \rho \frac{y \exp(-yf_x(\theta)) \nabla_\theta f_x(\theta)}{\|\exp(-yf_x(\theta)) \nabla_\theta f_x(\theta)\|} \tag{17}$$

$$= \theta - \rho y \frac{\nabla_\theta f_x(\theta)}{\|\nabla_\theta f_x(\theta)\|} \tag{18}$$

Importance weighting. We first aim to understand the ratio between the phantom and real importance weighting term $\tilde{\lambda}/\lambda$. We have,

$$\tilde{\lambda}/\lambda = \exp\left(-yf(\tilde{\theta})\right) / \exp\left(-yf(\theta)\right) \tag{19}$$

$$= \exp\left(-y(f(\tilde{\theta}) - f(\theta))\right) \tag{20}$$

which arises from plugging in the definition of the importance weighting term with exponential loss. For convenience, we will consider the log of the ratio,

$$\log\left(\tilde{\lambda}/\lambda\right) = -y(f(\tilde{\theta}) - f(\theta)). \tag{21}$$

The effect is based upon a first-order Taylor expansion of the logit function $f(\theta)$,

$$f(\tilde{\theta}) = f(\theta) - \rho\, y\, \nabla_\theta f(\theta)^\top \frac{\nabla_\theta f(\theta)}{\|\nabla_\theta f(\theta)\|} + \mathcal{O}\left(\rho^2\right) \tag{22}$$

$$\approx f(\theta) - \rho\, y\, \|\nabla_\theta f(\theta)\|\,. \tag{23}$$

We can plug this into the expression for the ratio of importance weighting terms, and we get,

$$\log\left(\tilde{\lambda}/\lambda\right) \approx -y\left(f(\theta) - \rho\, y\, \|\nabla_\theta f(\theta)\| - f(\theta)\right) \tag{24}$$

$$= \rho\, \|\nabla_\theta f(\theta)\| \tag{25}$$

$$\tag{26}$$

since $y^2 = 1$. We can see that the ratio of importance weighting terms is approximately proportional to the magnitude of the feature gradient $\|\nabla_\theta f(\theta)\|$.

For a better sense of the dynamics, we compute $\|\nabla_\theta f(\theta)\|$ for a few different architectures.

Example 1: LSAM. We consider the case of LSAM, in which only the last layer is perturbed to construct the phantom parameter. Since this is equivalent to considering the representation function ϕ to be a constant, we only need to compute the feature gradient norms with respect to the last layer, w. We have,

$$\|\nabla_w f(\theta)\| = \|\phi_\theta(x)\|. \tag{27}$$

This implies that the log importance re-weighting factor $\tilde{\lambda}/\lambda$ is proportional to the norm of the representation $\phi_\theta(x)$.

Example 2: Two-layer linear network. We consider the case of a two-layer linear network, with a single hidden layer. Let $f(\theta) = v^\top W x$ where $\theta = (v, W)$. For convenience, we will compute the squared norm of the feature gradient,

$$\|\nabla_\theta f(\theta)\|^2 = \|\nabla_v f(\theta)\|^2 + \|\nabla_W f(\theta)\|^2 \tag{28}$$

$$= \|Wx\|^2 + \|v\|^2\|x\|. \tag{29}$$

$$\tag{30}$$

Example 3: Multi-layer linear network. We consider the case of a multi-layer linear network. Let $f(\theta) = v^\top W_1 \cdots W_{L-1} x$ where $\theta = (v, W_1, \ldots, W_{L-1})$. For convenience, we will compute the squared norm of the feature gradient,

$$\|\nabla_\theta f(\theta)\|^2 = \|\nabla_v f(\theta)\|^2 + \sum_{i=1}^{L-1} \|\nabla_{W_i} f(\theta)\|^2 \tag{31}$$

$$= \|W_1 \cdots W_{L-1} x\|^2 + \sum_{i=1}^{L-1} \|W_1 \cdots W_{i-1} v\|^2 \|W_{i+1} \cdots W_{L-1} x\|^2 \tag{32}$$

$$= \sum_{j=1}^{L} a_j \|W_j \cdots W_{L-1} x\|^2 \tag{33}$$

where $a_j = \|W_1 \cdots W_{j-1} v\|^2$ is constant with respect to x. This means that the squared norm of the feature gradient is proportional to a weighted sum of the squared norms of the representations at each layer.

Example 4: Multi-layer perceptrons with ReLU. We can also compute the gradient of non-linear architectures, such as multi-layer perceptrons with ReLU activations. In particular, let $f(\theta) = v^\top \sigma(W_1 \sigma(W_2 \cdots \sigma(W_{L-1} x)))$ be defined by a sequence of layers with the ReLU activation function $\sigma(x) = \max(0, x)$. For a particular input, we can compute the output of a particular layer i as,

$$\sigma_i(x) = \sigma(W_i \sigma(W_{i+1} \cdots \sigma(W_{L-1} x))). \tag{34}$$

observe that for a given input x, if we define the matrix $A_i = \mathrm{diag}(\mathbb{I}\{\sigma_i(x) > 0\})$, then observe that the output of the network can be written as,

$$f(\theta) = v^\top A_1 W_1 A_2 W_2 \cdots A_{L-1} W_{L-1} x. \tag{35}$$

For convenience, we will compute the squared norm of the feature gradient,

$$\|\nabla_\theta f(\theta)\|^2 = \|\nabla_v f(\theta)\|^2 + \sum_{i=1}^{L-1} \|\nabla_{W_i} f(\theta)\|^2 \tag{36}$$

$$\begin{aligned} &= \|A_1 W_1 \cdots A_{L-1} W_{L-1}\|^2 \\ &\quad + \sum_{i=1}^{L-1} \|A_1 W_1 A_2 W_2 \cdots A_{i-1} W_{i-1} v\|^2 \|A_{i+1} W_{i+1} \cdots A_{L-1} W_{L-1} x\|^2 \end{aligned} \tag{37}$$

$$= \sum_{j=1}^{L} a_j \|A_j W_j \cdots A_{L-1} W_{L-1} x\|^2 \tag{38}$$

$$= \sum_{j=1}^{L} a_j \|\sigma_j(x)\|^2 \tag{39}$$

$$\tag{40}$$

where $a_j = \|A_1 W_1 \cdots A_{j-1} W_{j-1}\|^2$. Unlike the multi-layer linear case, this constant depends on x through A_j. This means that the squared norm of the feature gradient is proportional to a weighted sum of the squared norms of the representations at each layer, but where the weights depend on the input itself. Due to this, a general interpretation of the importance weighting effect is more difficult to obtain.

General multi-layer neural networks. In general, mutli-layer neural networks can introduce non-linearities that can make this gradient difficult to analyze. However, we suspect that the computation for mutli-layer linear networks will provide a sensible approximation for the feature gradient norm for multi-layer neural networks.

E DATA DISTRIBUTION

In this section, we describe the data distribution for the toy setup.

The data in the toy setup is a concatenation of two features that are independently drawn conditional on a label y. Before precisely defining the entire distribution, we will specify the feature distributions individually.

Easy feature distribution. We sample the easy feature, conditioned on the label $y \in \{-1, 1\}$, from a latent variable z. The latent variable z is sampled from the uniform distribution over the interval $[0, 1]$ if $y = 1$ and from a uniform distribution over the interval $[-1, 0]$ if $y = -1$. The easy feature is then defined as the 2-dimensional vector $[z, z]$, and multiplied by a fixed scale parameter a_{easy}. Together,

$$z_{\text{easy}} \sim \begin{cases} \mathcal{U}(0, 1) & \text{if } y = 1 \\ \mathcal{U}(-1, 0) & \text{if } y = -1 \end{cases} \tag{41}$$

$$x_{\text{easy}} = a_{\text{easy}} \cdot [z_{\text{easy}}, z_{\text{easy}}] \tag{42}$$

Hard feature distribution. Similarly, we sample the hard feature, conditioned on the label $y \in \{-1, 1\}$, from a latent variable z. The latent variable z^2 is sampled from the uniform distribution over the interval $[0, (2\pi\chi/360)^2]$ if $y = 1$ and from a uniform distribution over the interval $[-(2\pi\chi/360)^2, 0]$ if $y = -1$, where χ is a fixed scalar representing the complexity. The hard feature is then defined as the 2-dimensional vector $[-z \cos z, z \sin z]$, multiplied by a fixed scale parameter a_{hard}, and added to some 2-dimensional uniform noise $\eta \sim \mathcal{U}([0, 0.5] \times [0, 0.5])$. Together,

$$z_{\text{hard}}^2 \sim \begin{cases} \mathcal{U}(0, (2\pi\chi/360)^2) & \text{if } y = 1 \\ \mathcal{U}(-(2\pi\chi/360)^2, 0) & \text{if } y = -1 \end{cases} \tag{43}$$

$$x_{\text{hard}} = a_{\text{hard}} \cdot [-z_{\text{hard}} \cos z_{\text{hard}}, z_{\text{hard}} \sin z_{\text{hard}}] + \eta \tag{44}$$

Noise. For the variants that we present in the appendix, we add Gaussian noise $\varepsilon \sim \mathcal{N}(0, \sigma^2)$, feature label noise $u \sim R(p)$, and feature dropout noise $v \sim B(q)$, where $R(p)$ is the Rademacher distribution with probability p of -1 and probability $1-p$ of 1, and $B(q)$ is the Bernoulli distribution with probability q. The noise is added to the easy feature after it is scaled by a.

$$x_{\{\text{easy,hard}\}} = \varepsilon + u \cdot v \cdot a_{\{\text{easy,hard}\}} \cdot [z_{\{\text{easy,hard}\}}, z_{\{\text{easy,hard}\}}] \tag{45}$$

In general, we choose $a_{\text{easy}} = 2$ and $a_{\text{hard}} = 0.25$.

F DOMAINBED

In this section, we present comparison of SAM and SGD on DomainBed (**?**). DomainBed is a benchmark consisting of datasets with realistic domain shifts. Thus, the results on DomainBed are indicative of SAM's performance under real-world domain shifts in which the features that are useful for in-distribution classification may differ from the features that are useful for out-of-distribution classification. We compare SAM and SGD on three of the DomainBed datasets: OfficeHome, PACS, and VLCS.

F.1 DATASETS

OfficeHome is a dataset consisting of images from four domains: Art (A), Clipart (C), Product (P), and Real-World (R).

Table 2: Comparing the domain transfer performance of SAM and SGD on the DomainBed datasets.

OfficeHome

	Domain			
	A	C	P	R
SGD	0.687	0.716	0.853	0.777
SAM	**0.709**	**0.745**	**0.858**	**0.784**

PACS

	Domain			
	A	C	P	S
SGD	0.910	0.897	0.955	0.893
SAM	**0.914**	**0.921**	**0.964**	**0.908**

VLCS

	Domain			
	C	L	S	V
SGD	0.993	0.736	0.748	0.812
SAM	**0.996**	**0.748**	**0.776**	**0.830**

PACS is a dataset consisting of images from four domains: Art (A), Cartoon (C), Photo (P), and Sketch (S).

VLCS is a dataset consisting of images from four domains: Caltech101 (C), LabelMe (L), SUN09 (S), and VOC2007 (V).

We use the standard test/train splits of each of the datasets.

F.2 Experimental Setup

To evaluate domain transfer performance for a given dataset and domain, we train a classifier on the training set of all domains except the target domain. Following the setup of the main paper, we extract the quality of the representation by training a linear probe on the training dataset of the target domain.

More precisely, given the representation ϕ of the classifier that has been pre-trained on the other domains, we aim to minimize the following loss on the training set of the target domain:

$$\mathcal{L}(u) = \mathbb{E}_{(x,y)\sim D_{\text{train}}} \left[\ell(u^\top \phi(x), y) \right] \tag{46}$$

where D_{target} refers to the training distribution of the target domain, u is the linear probe we are optimizing, ℓ is cross-entropy loss, and x and y are the input and label of the training example. We evaluate by measuring the accuracy of the linear probe on the test set of the target domain.

We perform a hyperparameter sweep over $\rho \in \{0, 0.03, 0.05, 0.1\}$ and the learning rate $\eta \in \{0.005, 0.01, 0.02\}$. We validate on a small held-out validation set (10% of the training dataset) selected from the training dataset split. We report the results of the best hyperparameter setting on the test set of the target domain.

F.3 Results

We present the results of SAM and SGD on the three datasets in Table 2. We find that SAM outperforms SGD on all three datasets, confirming that SAM improves the representation quality of the neural network in a variety of domain transfer settings.

G Extended Figures

We provide extended versions of Figures 5 and 6 in Figures 8 and 9, respectively. These figures show the Lorenz curves for the real and phantom importance weights λ_i and $\tilde{\lambda}_i$ and the median importance weighting as a function of the contribution of each feature, respectively. We plot each result over multiple checkpoints over the duration of training.

Results. We observe that the trends discussed in Section 6.2 are consistent across the duration of training. The phantom importance weights $\tilde{\lambda}_i$ are more uniformly distributed than the real importance weights λ_i. Further, the median importance weighting is higher for points with a higher contribution of the easy feature than the hard feature.

H Variants of the toy setup

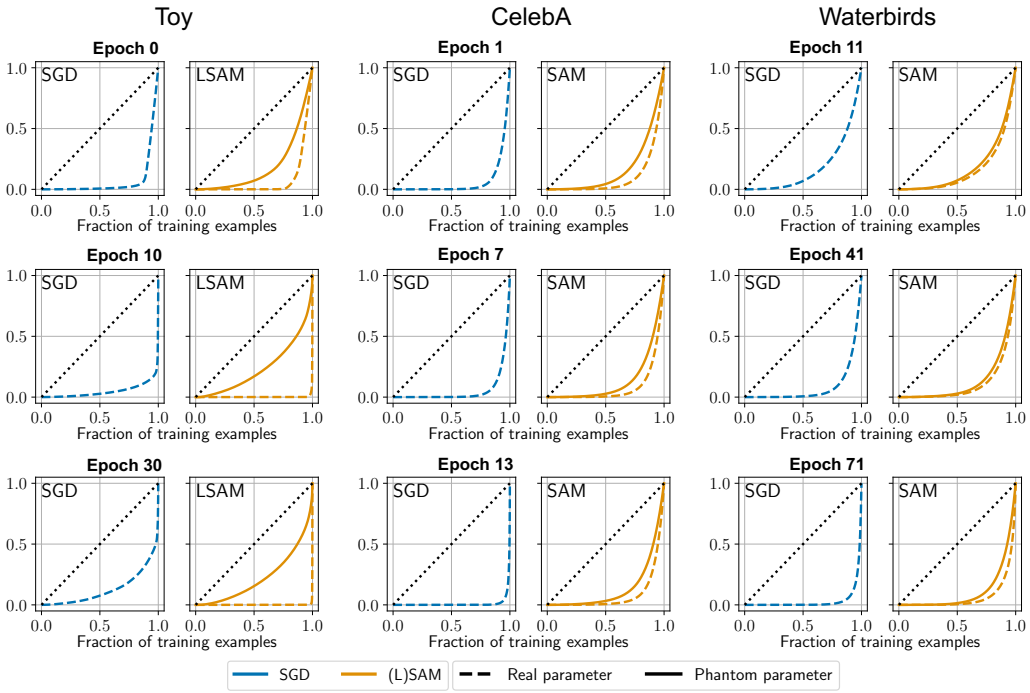

Figure 8: Extended version of Figure 5. Lorenz curves for the real and phantom importance weight λ_i and $\tilde{\lambda}_i$. The dotted diagonal line represents the Lorenz curve for a uniform distribution. The closer this curve is to this diagonal, the more equally the importance weights are spread. In blue, we plot the Lorenz curves for each SGD checkpoint. In orange, we plot the Lorenz curves for each LSAM checkpoint. The update step gradient is computed at real parameter for SGD, and the phantom parameter for SAM. We include the curves for the toy (left), CelebA (center), and Waterbirds (right). We observe that within the SAM checkpoints, the weights of points evaluated at the phantom parameter are closer to uniform than when evaluate at the real parameter. Further, in comparison to an SGD checkpoint, the phantom parameter of SAM weights points more uniformly. We plot each result over multiple checkpoints over the duration of training (epoch in bold).

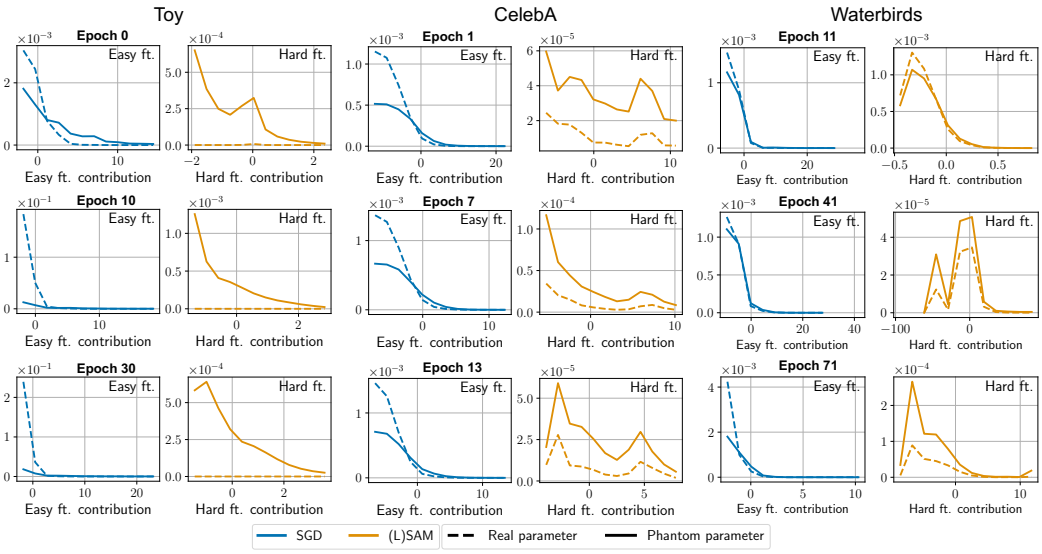

Figure 9: Extended version of Figure 6. Median importance weighting as a function of the contribution of each feature. We partition the data into bins based on the contribution of the easy and hard features ($yv_{\text{easy}}\Phi_{\text{easy}}$ and $yv_{\text{easy}}\Phi_{\text{hard}}$), as defined in Section 6.2. For each of these bins, we plot the median importance weight term λ_i for the points in the bin. We include the corresponding plots for the toy (top), CelebA (center), and Waterbirds (bottom). We plot each result over multiple checkpoints over the duration of training (epoch in bold).

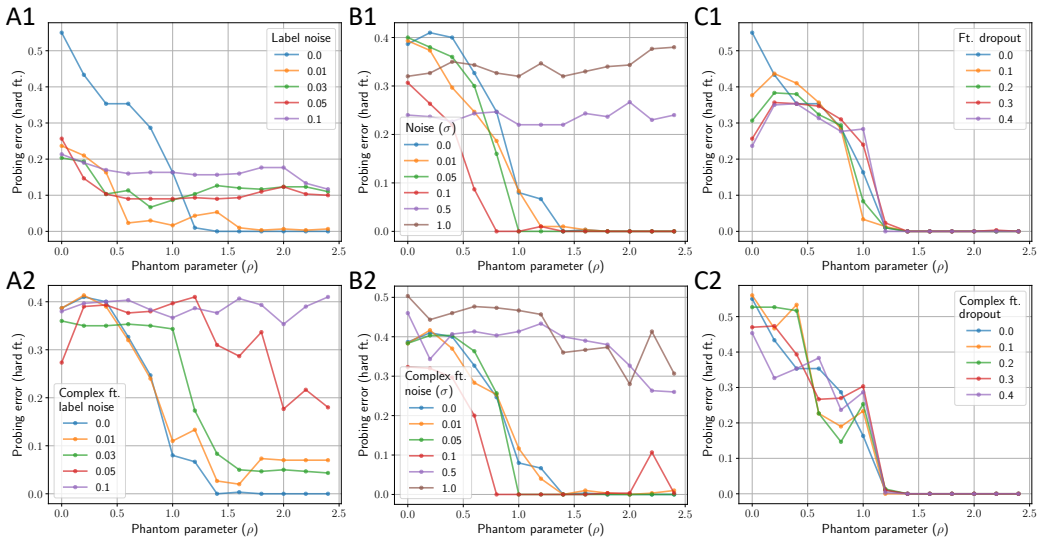

Figure 10: Variants of the toy setup to include noise in the feature distributions (compare with Figure 2). We include three types of noise distributions: (**A**) label noise, (**B**) Gaussian noise, and (**C**) feature dropout. For each type of noise, we apply the noise to: (**1**) both the simple and complex features, and (**2**) only the complex feature. We plot the probing error of the hard feature as a function of the LSAM phantom parameter ρ. Note that $\rho = 0$ corresponds to the baseline SGD model.

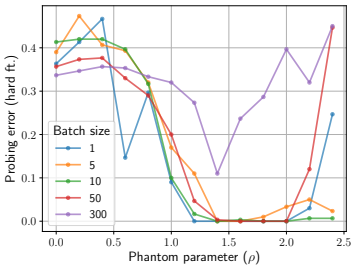

Figure 11: Variant of the toy setup in which we vary the batch size (compare with Figure 2). We plot the probing error of the hard feature as a function of the LSAM phantom parameter ρ. Note that $\rho = 0$ corresponds to the baseline SGD model.

In this section, we present variants of the toy setup to include noise in the feature distributions and to vary batch size. In general, our results are consistent with the results from the main paper. We observe that LSAM is robust to a wide range of noise distributions and batch sizes.

H.1 Noise distributions

In order to explore how LSAM behaves when features include noise, we train classifiers using the toy setup with three types of noise distributions.

Label noise. We add label noise by randomly flipping the labels of a fraction of the features when generating the training, as defined precisely in Section E. Note that this is different from how label noise is typically defined: we are not flipping the labels of the training examples, but rather the labels of the features. We vary the fraction of label noise from 0% to 10%.

Gaussian noise. We add Gaussian noise to the features, as defined precisely in Section E. We vary the standard deviation of the Gaussian noise from 0 to 1.0.

Feature dropout. We add feature dropout to the features, in which, with some probability, we replace features with 0, as defined precisely in Section E. We vary the probability of dropout from 0 to 0.4.

Hard-feature-only noise. For each type of noise, we also consider a variant in which the noise is only added to the hard feature.

In general, we find that LSAM improves the probing error of the hard feature in comparison to SGD for all types of noise (Figure 10). This suggests that the mechanisms of LSAM that improve feature representation quality are robust to noise in the feature distributions.

H.2 Batch size

We also vary the batch size in the toy setup. We plot the probing error of the hard feature as a function of the LSAM phantom parameter ρ in Figure 11. Note that $\rho = 0$ corresponds to the baseline SGD model. We observe that LSAM is generally robust to a wide range of batch sizes, and that the optimal value of ρ is similar across batch sizes. However, as batch size increases to be very large, the improvement of LSAM degrades, consistent with [CITE]. In addition, we observe that for large ρ, the probing error of the hard feature is more sensitive to batch size.