

Make the Most of Your Data: Changing the Training Data Distribution to Improve In-distribution Generalization Performance

Dang Nguyen¹ Paymon Haddad¹ Eric Gan¹ Baharan Mirzasoleiman¹

Abstract

Can we modify the training data distribution to encourage the underlying optimization method toward finding solutions with superior generalization performance on *in-distribution* data? In this work, we approach this question for the first time by comparing the inductive bias of gradient descent (GD) with that of sharpness-aware minimization (SAM). By studying a two-layer CNN, we prove that SAM learns easy and difficult features more uniformly, particularly in early epochs. That is, SAM is less susceptible to simplicity bias compared to GD. Based on this observation, we propose USEFUL, an algorithm that clusters examples based on the network output early in training and upsamples *examples with no easy features* to alleviate the pitfalls of the simplicity bias. We show empirically that modifying the training data distribution in this way effectively improves the generalization performance on the original data distribution when training with (S)GD by mimicking the training dynamics of SAM. Notably, we demonstrate that our method can be combined with SAM and existing data augmentation strategies to achieve, to the best of our knowledge, state-of-the-art performance for training ResNet18 on CIFAR10, STL10, CINIC10, Tiny-ImageNet; ResNet34 on CIFAR100; and VGG19 and DenseNet121 on CIFAR10.

works with improved structures (Liu et al., 2018; Zoph & Le, 2016; Pham et al., 2018) or higher-capacity (Nakkiran et al., 2021; Mei & Montanari, 2022). More recently, improving the quality of the training data has emerged as a popular avenue to improve generalization performance. Interestingly, higher-quality data can further improve the performance when larger models and better optimization methods are unable to do so (Hoffmann et al., 2022; Gadre et al., 2023). Recent efforts to improve the data quality have mainly focused on filtering irrelevant or harmful examples (Steinhardt et al., 2017; Li et al., 2020; Gadre et al., 2023). Nevertheless, it remains an open question if one can change the distribution of training data to further improve the in-distribution generalization performance of models trained on it.

At first glance, the above question may seem unnatural, as it disputes a fundamental assumption that training and test data should come from the same distribution (James et al., 2013). Under this assumption, minimizing the training loss generalizes well on the test data (Belkin et al., 2019). Indeed, machine learning models often experience a drastic drop in performance when employed under distribution shift (Quionero-Candela et al., 2009). Thus, improving the test performance by changing the distribution of training data may seem unexpected. At the same time, for overparameterized neural networks with more parameters than training data, there are many zero training error solutions, all global minima of the training objective, with different generalization performance (Gunasekar et al., 2017). With this in mind, one may still hope to change the data distribution to drive the optimization algorithm towards finding solutions with better generalization performance on the original data distribution.

In this work, we take the first steps towards answering the above question. To do so, we rely on recent results in non-convex optimization, showing the superior generalization performance of sharpness-aware-minimization (SAM) (Foret et al., 2020) over (stochastic) gradient descent (GD). SAM finds flatter local minima by simultaneously minimizing the loss value and loss sharpness. In doing so, it outperforms (S)GD and obtains state-of-the-art performance, at the expense of doubling the training time (Zheng et al., 2021; Du et al., 2021). Our key idea is that if one can change the training data distribution such that applying GD

1. Introduction

Training data is a key component of machine learning pipelines and directly impacts its performance. Over the last decade, there has been a large body of efforts concerned with improving learning from a given training dataset by designing more effective optimization methods (Foret et al., 2020; Kingma & Ba, 2014; Yao et al., 2021) or neural net-

¹Department of Computer Science, University of California, Los Angeles. Correspondence to: Dang Nguyen <dan-gnth@cs.ucla.edu>.

more closely simulates the training dynamics of SAM, then the new distribution may help GD converge toward more generalizable minima. However, it remains unclear how to modify the distribution to encourage this.

To address the above question, we first theoretically analyze the dynamics of training a two-layer convolutional neural network (CNN) with SAM and compare it with that of GD. We prove that SAM relies more on difficult features compared to GD, and that easy and difficult features contribute more *evenly* to the learning dynamics of SAM, particularly *early* in training. In other words, we show that SAM is less susceptible to *simplicity bias* than GD. This is interesting, as the simplicity bias of (S)GD towards learning simple features is conjectured to be the reason for the superior generalization performance of overparameterized neural networks as it provides implicit regularization properties (Neyshabur et al., 2014; Hu et al., 2020; Valle-Perez et al., 2018; Gunasekar et al., 2017; Belkin et al., 2019; Nakkiran et al., 2021).

Next, following our theoretical results, we formulate changing the distribution of a training dataset as upsampling examples that enhance the contribution of difficult features. To do so, we cluster examples based on their model outputs into two groups *early* in training. Then to mitigate simplicity bias, we upsample the cluster with the larger average loss, which we show contains examples with no easy features. This effectively alleviates the simplicity bias early in training and consequently improves generalization performance. We empirically confirm that the best epoch for clustering is when the reduction in the training error starts to diminish.

We showcase the effectiveness of our approach in alleviating the simplicity bias and improving the generalization performance via extensive experiments on CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009). In particular, we show that our method, UpSample Early For Uniform Learning, (USEFUL), further improves the performance of gradient-based optimization and data augmentation methods. First, we show that despite being relatively lightweight, USEFUL effectively improves the generalization performance of SGD. Additionally, we show that applying SAM and TrivialAugment (TA) (Müller & Hutter, 2021) to the modified data distribution obtained by our method achieves, to the best of our knowledge, *state-of-the-art* accuracy for image classification for training ResNet18 on CIFAR10, STL10, CINIC10, Tiny-ImageNet; ResNet34 on CIFAR100; and VGG19 and DenseNet121 on CIFAR10.

Contributions. We make the following contributions:

- We theoretically characterize the training dynamics of GD and SAM when learning a two-layer nonlinear CNN and show that SAM relies more on difficult features, particularly early in training, and is consequently less reliant on simplicity bias to learn.

- We propose UpSample Early For Uniform Learning, (USEFUL), to cluster examples based on early model output and upsample the cluster with higher loss to reduce the simplicity bias during training.
- Our experiments highlight the superior performance of USEFUL. In particular, our method improves the performance of SGD and applied with SAM and TA obtains *state-of-the-art* performance for a variety of model architectures and datasets.

2. Related Works

Sharpness-aware-minimization (SAM). Motivated by the generalization advantages of flat local minima, sharpness-aware minimization (SAM) was concurrently proposed in (Foret et al., 2020; Zheng et al., 2021) to minimize the training loss at the worst perturbed direction from the current parameters. SAM has been shown to obtain state-of-the-art on a variety of tasks (Foret et al., 2020). Besides, SAM has been also shown to be beneficial in other settings, including label noise (Foret et al., 2020; Zheng et al., 2021), and domain generalization (Cha et al., 2021; Wang et al., 2023).

There have been recent efforts to understand the training dynamics of SAM and why it contributes to the model generalization. The most popular explanation is based on the study of Hessian spectra both empirically (Foret et al., 2020; Kaur et al., 2023) and theoretically (Wen et al., 2022; Bartlett et al., 2023). Other explanations include implicit bias towards sparse solutions in diagonal linear networks (Andriushchenko & Flammarion, 2022) and benign overfitting (Chen et al., 2023). More recently Springer et al. (2024) suggested that SAM promotes diverse feature learning by empirically studying a simplified version of SAM which only perturbs the last layer. They showed that SAM upscales the last layer’s weights to induce feature diversity.

In our work, we show that training data distribution can be modified to obtain a solution with similar properties to SAM, such as being sparser, having a smaller largest Hessian eigenvalue, and better in-distribution generalization performance.

Simplicity bias. SGD has an inductive bias towards learning simpler solutions with minimum norm (Gunasekar et al., 2017). Indeed, it is empirically observed (Kalimeris et al., 2019) and theoretically proved (Hu et al., 2020) that SGD learns linear functions in the early training phase and more complex functions later in training. The simplicity bias of SGD has been long conjectured to be the reason for the superior generalization performance of overparameterized neural networks, by providing a form of capacity control or implicit regularization (Neyshabur et al., 2014; Hermann & Lampinen, 2020; Shah et al., 2020; Pezeshki et al., 2021).

In our work, we study a two-layer nonlinear CNN and rigorously show that SAM is less susceptible to simplicity bias

than GD, in particular early in training, which contributes to its superior performance. Then, we show that training data distribution can be modified to reduce the simplicity bias and improve the in-distribution generalization performance.

Distinction from Existing Settings. Our work is distinct from the literature on (1) distribution shift, (2) improving the convergence of gradient methods, and (3) data filtering:

(1) *Distribution Shift.* Unlike the literature on distribution shift and shortcut learning (Sagawa et al., 2019; Kirichenko et al., 2022; Deng et al., 2023; Puli et al., 2023), we *do not* assume existence of strong spurious correlations in the training data or a shift between training and test distribution. Our theoretical and empirical results focus on *in-distribution* generalization, where training and test distributions are the same. In Appendix D.5 we empirically show the benefits of our method to distribution shift, but we emphasize that this is not the focus of our study and we leave this direction to future work.

(2) *Improving Convergence.* There is a body of work on speeding up convergence of gradient descent to find the *same solution* faster. Such methods iteratively sample or reweight examples based on loss or gradient norm during training (Zhao & Zhang, 2015; Katharopoulos & Fleuret, 2018; Johnson & Guestrin, 2018; El Hanchi et al., 2022). In contrast, our work does not intend to speed up training to find the same solution faster, but intends to find a *more generalizable solution*, by mimicking dynamics of SAM.

(3) *Data Filtering Methods.* Data filtering methods aim to improve the performance by identifying and discard or mitigate the effect of noisy labeled (Li et al., 2020), domain mismatched (Gadre et al., 2023), redundant (Lee et al., 2021; Raffel et al., 2020; Abbas et al., 2023), or adversarial examples crafted by data poisoning attacks (Steinhardt et al., 2017). In contrast, we assume a *clean* training data and no mismatch between training and test distribution. Indeed, our work can be applied to a filtered training data to further improve the generalization performance.

3. Theoretical Analysis: SAM Learns Easy/Difficult Features More Evenly

In this section, we analyze and compare feature learning mechanism of SAM and SGD. First, we introduce our theoretical settings including data distribution and neural network model in Section 3.1. We then revisit the update rules of GD and SAM in Section 3.2 before presenting our theoretical results in Section 3.3.

3.1. Theoretical Settings

Notation. We use lowercase letters, lowercase boldface letters, and uppercase boldface letters to respectively de-

note scalars (a), vectors ($\mathbf{v}$), and matrices ($\mathbf{W}$). For a vector $\mathbf{v}$, we use $\|\mathbf{v}\|_2$ to denote its Euclidean norm. Given two sequence $\{x_n\}$ and $\{y_n\}$, we denote $x_n = O(y_n)$ if $|x_n| \leq C_1|y_n|$ for some absolute positive constant C_1 , $x_n = \Omega(y_n)$ if $|x_n| \geq C_2|y_n|$ for some absolute positive constant C_2 , and $x_n = \Theta(y_n)$ if $C_3|y_n| \leq |x_n| \leq C_4|y_n|$ for some absolute constant $C_3, C_4 > 0$. In addition, we use $\tilde{O}(\cdot)$, $\tilde{\Omega}(\cdot)$, and $\tilde{\Theta}(\cdot)$ to hide logarithmic factors in these notations. Furthermore, we denote $x_n = \text{poly}(y_n)$ if $x_n = O(y_n^D)$ for some positive constant D , and $x_n = \text{polylog}(y_n)$ if $x_n = \text{poly}(\log(y_n))$.

Data distribution. We use a popular data distribution in recent works on feature learning (Allen-Zhu & Li, 2020; Chen et al., 2022; Jelassi & Li, 2022; Cao et al., 2022; Kou et al., 2023; Deng et al., 2023; Chen et al., 2023) to represent data as a combination of easy, difficult, and noise patches. Additionally, we introduce a probability α to control the frequency of easy features in the data distribution. Typically in real-world datasets, a small fraction of the examples in the train dataset cannot be learned using only easy features (as evidenced by simplicity bias), so we use α to model this.

Definition 3.1 (Data distribution). A data point $(\mathbf{x}, y) \in (\mathbb{R}^d)^P \times \{\pm 1\}$ is generated from the distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ as follows.

- Uniformly generate the label $y \in \{\pm 1\}$.
- Generate $\mathbf{x}$ as a collection of P patches: $\mathbf{x} = (\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(P)}) \in (\mathbb{R}^d)^P$, where
 - **Difficult feature.** One and only one patch is given by $\beta_d \cdot y \cdot \mathbf{v}_d$ with $\|\mathbf{v}_d\|_2 = 1$, $\langle \mathbf{v}_e, \mathbf{v}_d \rangle = 0$, and $0 \leq \beta_d < \beta_e \in \mathbb{R}$.
 - **Easy feature.** One and only one patch is given by $\beta_e \cdot y \cdot \mathbf{v}_e$ with $\|\mathbf{v}_e\|_2 = 1$ with a probability $\alpha \leq 1$. With a probability of $1 - \alpha$, this patch is masked, i.e. $\mathbf{0}$.
 - **Random noise.** The rest of $P - 2$ patches are Gaussian noise $\boldsymbol{\xi}$ that are independently drawn from $N(0, (\sigma_p^2/d) \cdot \mathbf{I}_d)$ with σ_p as an absolute constant.

For simplicity, we assume $P = 3$, and the noisy patch together with two features form an orthogonal set. Coefficients β_e and β_d characterize the feature strength in our data model. A larger coefficient means that the corresponding feature is more noticeable, thus being easily learned by GD. While we do not capture other measures of difficulty such as shape in our theory, our conclusions are still valid as we will explain in Section 4 (Step 1).

Two-layer nonlinear CNN. To model modern state-of-the-art architectures, we analyze a two-layer nonlinear CNN which is also used in (Chen et al., 2022; Jelassi & Li, 2022; Cao et al., 2022; Kou et al., 2023; Deng et al., 2023). The

advantage of the CNN structure over linear models is that it can handle a data distribution that does not require a fixed position of patches as defined above. Formally,

$$f(\mathbf{x}; \mathbf{W}) = \sum_{j \in [J]} \sum_{p=1}^P \sigma(\langle \mathbf{w}_j, \mathbf{x}^{(p)} \rangle), \quad (1)$$

where $\mathbf{w}_j \in \mathbb{R}^d$ is the weight vector of the j -th filter, J is the number of filters (neurons) of the network, and $\sigma(z) = z^3$ is the activation function which is the main source of non-linearity. $\mathbf{W} = [\mathbf{w}_1, \dots, \mathbf{w}_J] \in \mathbb{R}^{d \times J}$ denotes the weight matrix of the CNN. Following (Jelassi & Li, 2022; Cao et al., 2022; Deng et al., 2023), we assume a mild overparameterization of the CNN with $J = \text{polylog}(d)$. We initialize $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$, where $\sigma_0^2 = \text{polylog}(d)/d$.

3.2. Empirical Risk Minimization

Consider a training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ in which each data point is generated from the data distribution in Definition 3.1. The empirical loss function of a model $f(\mathbf{x}; \mathbf{W})$ reads

$$\mathcal{L}(\mathbf{W}) = \frac{1}{N} \sum_{i=1}^N l(y_i f(\mathbf{x}_i; \mathbf{W})), \quad (2)$$

where l is the logistic loss defined as $l(z) = \log(1 + \exp(-z))$. The solution $\mathbf{W}^*$ of the empirical risk minimization (ERM) minimizes the above loss, i.e., $\mathbf{W}^* := \arg \min_{\mathbf{W}} \mathcal{L}(\mathbf{W})$. Typically, ERM is solved using gradient descent (GD) algorithms. For example, the update rule at iteration t of GD with learning rate $\eta > 0$ reads

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{W}^{(t)}). \quad (3)$$

SAM. To find solutions with better generalization performance, Foret et al. (2020) proposed the N -SAM algorithm that minimizes both loss and curvature. SAM's update rule at iteration t reads

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \eta \nabla \mathcal{L}(\mathbf{W}^{(t)} + \rho^{(t)} \nabla \mathcal{L}(\mathbf{W}^{(t)})), \quad (4)$$

where $\rho^{(t)} = \rho > 0$ is the inner step size which is usually normalized by gradient norm, i.e., $\rho^{(t)} = \rho / \|\nabla \mathcal{L}(\mathbf{W}^{(t)})\|_F$.

3.3. Comparing Learning Between Easy/Difficult Features for ERM and SAM

In this subsection, we present our theoretical results on the training dynamics of the two-layer nonlinear CNN using GD and SAM. We characterize the learning speed of easy and difficult features by studying the growth of the model outputs before the activation function, i.e., $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle$ and $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle$. We first prove in early training, both GD and SAM learn easy features faster than difficult ones. Then, we show SAM does not over-rely on easy features and consequently learns easy and difficult features more uniformly.

Theorem 3.2 (GD Feature Learning). Consider training a two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ on the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ with $\alpha^{1/3} \beta_e > \beta_d$. Assuming that the learning rate η is sufficiently small, after training with GD in Eq. (3) for T_0 iterations where

$$T_0 = \frac{\tilde{\Theta}(1)}{\eta \beta_e^3 \sigma_0} + \tilde{\Theta}(1) \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil, \quad (5)$$

for all $j \in [J]$ and $t \in [0, T_0)$, we have

$$\tilde{\Theta}(\eta) \alpha \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 = \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle, \quad (6)$$

$$\tilde{\Theta}(\eta) \beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle. \quad (7)$$

After training for T_0 iterations, with high probability, the model has the following properties: (1) it learns the easy feature $\mathbf{v}_e$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$; (2) it does not learn the difficult feature $\mathbf{v}_d$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.

Theorem 3.3 (SAM Feature Learning). Consider training a two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ on the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ with $\alpha^{1/3} \beta_e > \beta_d$. Assuming that the learning rate η and the perturbation radius ρ are sufficiently small, after training with SAM in Eq. (3) for T_0 iterations where

$$T_0 = \frac{\tilde{\Theta}(\sigma_0)}{\eta \beta_e^3 (\sigma_0 - \rho)^2} + \frac{\tilde{\Theta}(\sigma_0^2)}{(\sigma_0 - \rho)^2} \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil, \quad (8)$$

for all $j \in [J]$ and $t \in [0, T_0)$, we have

$$\begin{aligned} \tilde{\Theta}(\eta) \alpha \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 &= \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle \\ &\leq \tilde{\Theta}(\eta) \alpha \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2, \end{aligned} \quad (9)$$

$$\begin{aligned} \tilde{\Theta}(\eta) \beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 &= \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \\ &\leq \tilde{\Theta}(\eta) \beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2, \end{aligned} \quad (10)$$

where $\mathbf{w}_{j,\epsilon}^{(t)} = \mathbf{w}_j^{(t)} + \rho^{(t)} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)})$ is the perturbed weight in SAM. After training for T_0 iterations, with high probability, the model has the following properties: (1) it learns the easy feature $\mathbf{v}_e$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$; (2) it does not learn the difficult feature $\mathbf{v}_d$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.

The detailed proof of Theorems 3.2 and 3.3 are deferred to Appendices A.1 and A.2.

Discussion. Note that a larger value of $\langle \mathbf{w}_j^{(t)}, \mathbf{v} \rangle$ for $\mathbf{v} \in \{\mathbf{v}_e, \mathbf{v}_d\}$ indicates better learning of the feature vector $\mathbf{v}$ by neuron $\mathbf{w}_j$ at iteration t . From the above two theorems, the growth rate of the easy feature is significantly faster than that of the difficult feature. Because the model output of data point $\mathbf{x}_i$ at iteration t is given by $f(\mathbf{x}_i; \mathbf{W}) =$

$\sum_{j \in [J]} (y\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 + y\beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 + \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^3)$, the term $\beta_e^3 \max_{j \in [J]} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3$ greatly affects the model prediction. However, a small portion $(1 - \alpha)$ of the dataset does not have the easy feature, thus, the model cannot rely on the easy feature to predict this group. If the difficult feature is not learned, i.e., indistinguishable from noise, the prediction is random, hurting the model performance. Therefore, it necessitates the learning of the difficult feature to improve the generalization. In addition, comparing Equations 5 and 8, we can see that SAM learns the easy features later than GD. Particularly, if we remove the approximation notations, we have the following inequality $\frac{1}{\eta\beta_e^3\sigma_0} + \left\lceil \frac{-\log(\sigma_0\beta_e)}{\log(2)} \right\rceil \leq \frac{\sigma_0}{\eta\beta_e^3(\sigma_0-\rho)^2} + \frac{\sigma_0^2}{(\sigma_0-\rho)^2} \left\lceil \frac{-\log(\sigma_0\beta_e)}{\log(2)} \right\rceil$, which holds due to our Assumption A.1 about weight initialization in Appendix A.1, i.e., $(\sigma_0 \geq \rho \geq 0)$. We will confirm this results in our toy experiment.

Next, we show that SAM learns easy and difficult features more evenly. To ease the notation, we denote by $G_e^{(t)} = \max_{j \in [J]} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle$ and $G_d^{(t)} = \max_{j \in [J]} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle$ the alignment of model weights with easy and difficult features, respectively, when training with GD. Similarly, we denote by $S_e^{(t)}$ and $S_d^{(t)}$ the alignment of model weights with easy and difficult features, when training with SAM.

Theorem 3.4 (SAM learns features more evenly than GD). *We consider training a two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ on the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ with $\alpha^{1/3}\beta_e > \beta_d$. We assume that the learning rate η and the perturbation radius ρ are sufficiently small. Starting from the same initialization, the growth of easy and difficult features in SAM is more balanced than that in SGD, i.e., for every iteration $t \in [1, T_0]$, we have*

$$S_e^{(t)} - S_d^{(t)} < G_e^{(t)} - G_d^{(t)}. \quad (11)$$

We prove Theorem 3.4 by induction in Appendix A.2 and back it by toy experiments in Section 5.1.

Discussion. Intuitively, our proof is based on the fact that the difference between the growth of easy and difficult features in SAM (the LHS in Equations 9 and 10) is smaller than that of GD (the LHS in Equations 6 and 7). Therefore, starting from the same initialization, the difficult feature contributes relatively more to the model prediction in SAM than it does in SGD. The difficult feature then can benefit the learning of SAM, by reducing its overreliance on the easy features. We note that as NNs are nonlinear, a small change in the output can actually result in a big change in the model and its performance. Even in the extreme setting when two features have identical strength and the easy feature exists in all examples, i.e., $\beta_e = \beta_d = \alpha = 1$, the gap in Equation 11 is significant as shown in Figure 6 in Appendix D.

From the recursions in Theorems 3.2 and 3.3, it is intuitive that we can make the model learn more from the difficult feature by increasing the value of β_d . Based on this intuition, we have the following theorem.

Theorem 3.5 (One-shot upsampling). *Consider the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ using the two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. Assume that the noise is sufficiently small and the easy feature strength is significantly larger $\alpha^{1/3}\beta_e > \beta_d$. From any iteration t during early training, we have the following results:*

1. *The difficult feature has a larger contribution to the normalized gradient of the one-step SAM update, compared to that of GD.*
2. *Amplifying the strength of the difficult feature increases its contribution to the normalized gradients of GD and SAM.*
3. *There exists an upsampling factor k such that the normalized gradient of the one-step GD update on $\mathcal{D}(\beta_e, k\beta_d, \alpha)$ recovers the normalized gradient of the one-step SAM update on $\mathcal{D}(\beta_e, \beta_d, \alpha)$.*

Discussion. The proof of Theorem 3.5 is given in Appendix A.4. Based on this theorem, we can stimulate the training dynamics of SAM by training GD on a new dataset $\mathcal{D}(\beta_e, \beta'_d, \alpha)$ with a larger difficulty strength $\beta'_d > \beta_d$. However, the value of coefficient β'_d varies by the model weights and gradient at each iteration t .

Remark. Intuitively, Theorem 3.5 implies that by descending over a flatter trajectory, SAM learns difficult features earlier in training. While the largest difference between feature learning of SAM and (S)GD is attributed to early training dynamics (due to the simplicity bias of (S)GD and its largest contribution early in training), SAM learns features at a more uniform speed during the entire training. This effect is, however, difficult to theoretically characterize exactly. In addition, easier features will continue to dominate during the training and are learned in the order of their learning difficulty. Therefore, learning features at a more uniform speed (via SAM and USEFUL) help yield flatter minima with better generalization performance, as we will also confirm empirically. Finally, we note that while SAM learns easy and difficult features at a more uniform speed, it still suffers from simplicity bias (although less than GD) and learns easier features earlier as evidenced in our Theorem 3.3. In the next Section, we will discuss how modifying the data distribution help GD and SAM find better solutions.

4. Method

Motivated by our theoretical results, we aim to amplify the strength of the difficult features in the training data. In

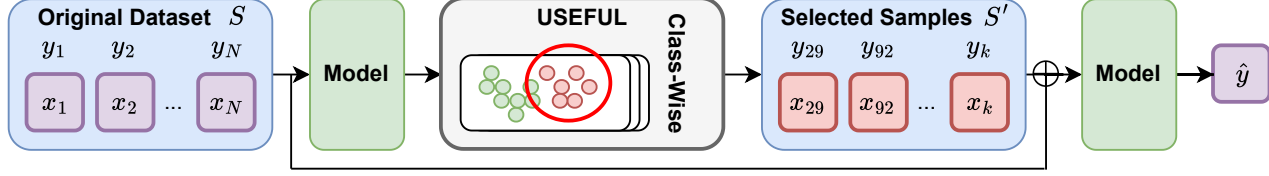


Figure 1: **Method overview.** USEFUL first trains the model for a few epochs t , which in practice is around 5-10% of the total training epochs. It then clusters examples in every class into 2 groups and upsamples examples in the cluster with higher average loss. Finally, the base model is retrained from scratch on the modified data distribution.

doing so, we increase the contribution of difficult features to learning and drive the network to learn easy and difficult features more uniformly in order to reduce simplicity bias.

Step 1: Identifying examples without easy features.

When a feature is learned by the network (i.e. has a large inner product with the weights), the model relies on that feature to derive its output towards the right one-hot encoded label vector for all examples containing that feature. As shown in Theorems 3.2 and 3.3, easy features are learned early in training, and thus the model outputs for examples containing *at least one easy features* are highly separable from the rest of examples, once the easy features are learned. Based on this observation, we partition examples in each class based on the model’s output into two clusters early in training. Specifically, in line with Theorems 3.2 and 3.3, we group examples by their last-layer activations using k -means clustering per class. Formally, for examples in every class with $y_j = c$, we find two clusters C_1, C_2 as follows:

$$\arg \min_C \sum_{i \in \{1,2\}} \sum_{\substack{y_j=c, \\ j \in C_i}} \|f(x_j; \mathbf{W}^{(t)}) - \mu_i\|^2, \quad (12)$$

where μ_i is the center of cluster S_i . The cluster with a lower average loss contains examples with at least one easy feature and the cluster with a higher average loss contains examples without any easy feature.

The choice of clustering. Some easy features may not be *fully* learned early in training. Hence, examples containing easy features may not necessarily have the right prediction, thus metrics such as misclassification cannot separate examples with easy features accurately. In addition, and there may not be a hard cut-off on the loss to separate examples. Nevertheless, examples with easy features are separable from other examples early in training, by clustering the model’s output. We confirm the advantage of clustering over other methods in Table 6 in our ablation studies.

Easy features in real-world data. In our work, we regard easy features as those that are learned early in training, due to simplicity bias of gradient methods. While our data model only models difficulty of features by their magnitude, in practice our method captures various measures of fea-

ture difficulty including shape, etc., by separating examples based on model’s output early in training.

Step 2: Upsampling examples without easy features.

Next, we upsample examples in the cluster with a higher average loss, as they do not contain any easy feature. This increases the strength of difficult features and encourages the model to learn easy and difficult features at a more balanced manner. Thus, it improves the in-distribution performance by making training dynamics more similar to SAM based on Theorem 3.5. As discussed earlier, the number of times we upsample these examples should change based on the model weight at each iteration. Nevertheless, we empirically confirm that finding difficult examples at an *early* training iteration and upsampling them by a factor of $k = 2$ effectively improves the generalization performance. We note that USEFUL upsamples examples only once and restart training on the modified but fix distribution. This is in contrast to methods that dynamically sample or weigh examples during training.

When to separate the examples. It is crucial to separate examples *early* in training, to accurately identify examples that contribute the most to simplicity bias. We empirically verify the intuition that the optimal epoch t to separate examples is when the change in training error starts to shrink as visualized in Figure 11a. More details can be found in Appendices C.2 and D.6.

The workflow of USEFUL is illustrated in Figure 1.

5. Experiments

Outline. In this section, we first describe our experimental settings. Then, in Section 5.1, we empirically validate our theoretical results on toy datasets. We then evaluate the performance of USEFUL on different datasets in Section 5.2 and different model architectures in Section 5.3. In addition, Section 5.4 highlights the advantages of USEFUL over random upsampling. Furthermore, we show that USEFUL shares several properties with SAM in Section 5.5. Additional experimental results are deferred to Appendix D where we show that USEFUL generalizes to other SAM variants, the OOD and transfer learning settings. We further

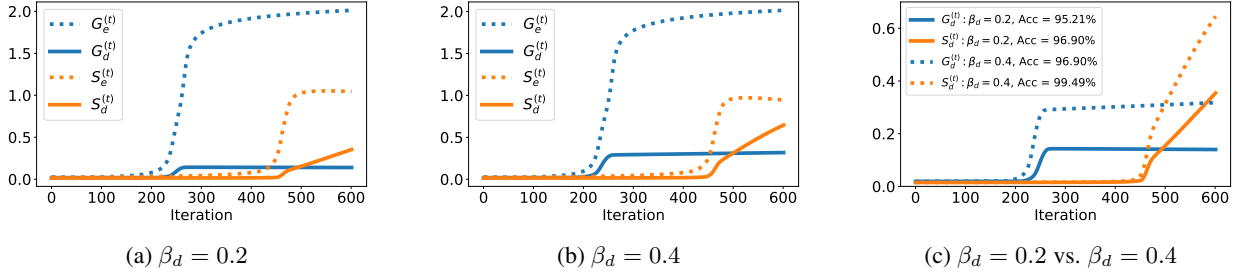


Figure 2: **Training dynamics of GD (blue) vs. SAM (orange) on toy datasets.** The toy datasets is generated from the distribution in Definition 3.1 with different values of β_d while fixing $\beta_e = 1$ and $\alpha = 0.9$. The dotted and solid lines denote the alignment (i.e., inner product) of easy (v_e) and difficult (v_d) features with the model weight ($w_j^{(t)}$). (a) and (b) GD and SAM learn the easy feature faster. In particular, GD learns the easy feature very early in training. (c) Test accuracy of GD and SAM increases when the strength of the difficult feature is larger.

conduct ablation studies on the effect of our data selection strategy, training batch size, learning rate, upsampling factor, and separating epoch in Appendix D.6.

Settings. We used common benchmark datasets for image classification including CIFAR10, CIFAR100 (Krizhevsky et al., 2009), STL10 (Coates et al., 2011), CINIC10 (Darló et al., 2018), and Tiny-ImageNet (Le & Yang, 2015). Both CINIC10 and Tiny ImageNet are large-scale datasets that contain images from the ImageNet dataset (Deng et al., 2009). We trained ResNet18 on all datasets except for CIFAR100 on which we trained ResNet34. We closely followed the setting from (Andriushchenko & Flammarion, 2022) in which our models are trained for 200 epochs with a batch size of 128. We used SGD with the momentum parameter of 0.9 and set weight decay to 0.0005. We also fixed $\rho = 0.1$ for SAM in all experiments unless explicitly stated. We used a linear learning rate schedule starting at 0.1 and decay by a factor of 10 once at epoch 100 and again at epoch 150. More details are given in Appendix C.

5.1. Toy Datasets

Datasets. Similar to (Deng et al., 2023), our toy dataset consists of training and test sets, each containing 10,000 examples generated from the data distribution defined in 3.1 with dimension $d = 50$ and $P = 3$. For other parameters, we set $\beta_e = 1, \beta_d = 0.2, \alpha = 0.9$, and $\sigma_p/\sqrt{d} = 0.125$. We also consider an additional scenario with larger $\beta_d = 0.4$. We shuffle the order of patches randomly to confirm that our theoretical results hold with an arbitrary order of patches.

Training. We used the two-layer nonlinear CNN in Section 3.1 with the number of filters $J = 40$. For training using gradient descent, we set the learning rate to $\eta = 0.1$ and did not use momentum. For SAM, we used the same base GD optimizer and chose a small value of the inner step $\rho = 0.02$ which is smaller than those in other experiments to satisfy the constraint in Theorem 3.4. We trained the model for 600 iterations till convergence for both GD and SAM.

Results. Figure 2a illustrates that both GD (blue) and SAM (orange) learn the easy feature faster, indicated by the dotted lines corresponding to G_e, S_e being above the solid lines corresponding to G_d, S_d . In particular, the blue dotted line accelerates quickly at around epoch 250 while the orange dotted line increases drastically much later at around epoch 450. That is: **(1) GD learns the easy feature very early in training.** This is well-aligned with our Theorems 3.2 and 3.3 and their discussion. Furthermore, the gap between contribution of easy and difficult features towards the model output in SAM ($S_e^{(t)} - S_d^{(t)}$) is much smaller than that of GD ($G_e^{(t)} - G_d^{(t)}$). That is: **(2) easy and difficult features are learned more evenly in SAM.** This validates our theoretical results in Theorem 3.4. From around epoch 500 onwards, the contribution of the difficult feature in SAM surpasses the level of that in GD while the contribution of the easy feature in SAM is still lower than the counterpart in GD. When increasing the difficult feature strength β_d from 0.2 to 0.4 in Figure 2b, the same conclusion for the growth speed of easy and difficult features holds. Notably, there is a clear increase in the classification accuracy of the model trained with either GD or SAM by increasing β_d , as can be seen in Figure 2c. That is: **(3) amplifying the strength of the difficult feature improves the generalization performance.** Effectively, this enables the model successfully predict examples in which the easy feature is missing.

5.2. USEFUL is Effective across Datasets

Figure 3 illustrates the performance of models trained with SGD and SAM on original vs modified data distribution by USEFUL. We see that USEFUL effectively reduces the test classification error of both SGD and SAM. Interestingly, reducing the simplicity bias further improves SAM’s generalization performance. Notably, on the STL10 dataset, USEFUL boosts the performance of SGD to surpass that of SAM. Stacking strong augmentation methods such as TrivialAugment (Müller & Hutter, 2021) further improves the performance, achieving state-of-the-art results for ResNet

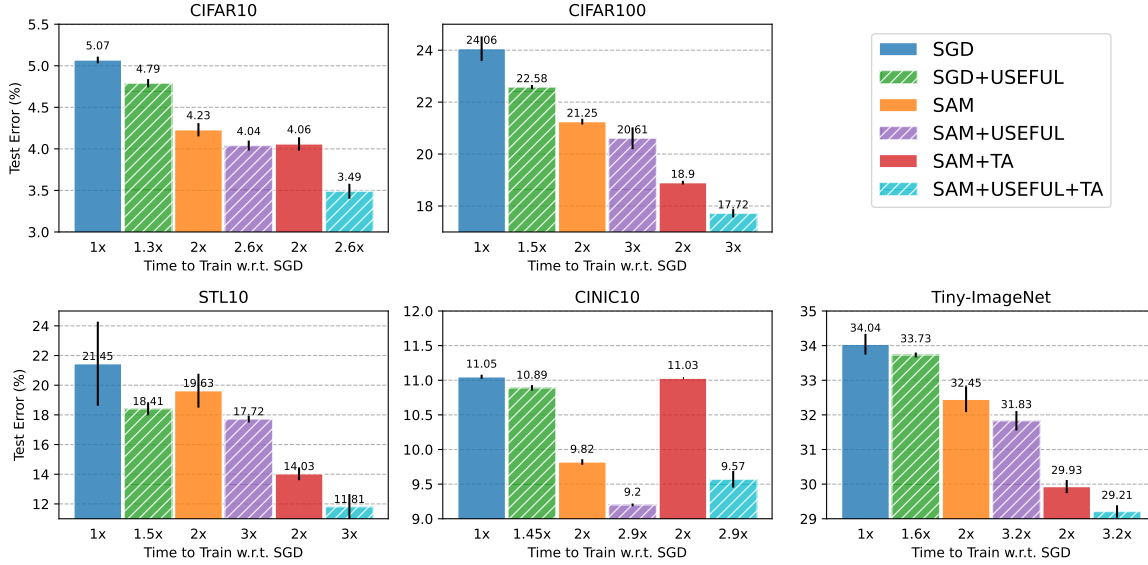


Figure 3: **Test classification error of ResNet18 on CIFAR10, STL10, TinyImageNet and ResNet34 on CIFAR100.** The number below each bar indicates the approximated cost to train the model and the tick on top shows the standard deviation over three runs. USEFUL enhances the performance of SGD and SAM on all 5 datasets. Leveraging TrivialAugment (TA) further boosts SAM’s performance (except for CINIC10). Remarkably, USEFUL consistently boosts the performance across all scenarios and achieves (to our knowledge) SOTA performance for ResNet18 and ResNet34 on the selected datasets when combined with SAM and TA.

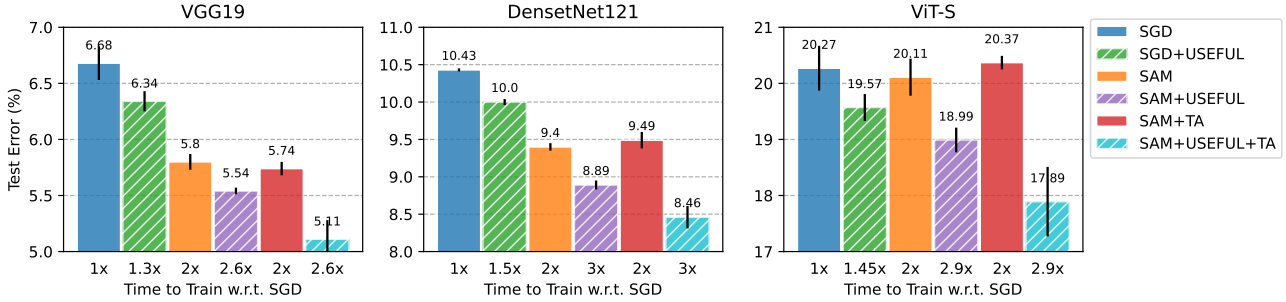


Figure 4: **Test classification errors of different architectures on CIFAR10.** The number below each bar indicates the approximated cost to train the model and the tick on top shows the standard deviation over three runs. USEFUL improves the performance of SGD and SAM when training with different architectures. Leveraging TrivialAugment (TA) further boosts SAM’s capabilities. The results for 3-layer MLP can be found in Figure 7.

on all datasets. The percentages of examples found for up-sampling by USEFUL for CIFAR10, CIFAR100, STL10, CINIC10, and Tiny-ImageNet are roughly 30%, 50%, 50%, 45%, and 60%, respectively. Thus, training SGD on the modified data distribution only incurs a cost of 1.3x, 1.5x, 1.5x, 1.45x, and 1.6x compared to 2x of SAM.

5.3. USEFUL is Effective across Architectures & Settings

Model architectures: CNN, ViT, MLP. Next, we confirm the versatility of our method, by applying it to different model architectures including 3-layer MLP, CNNs (ResNet18, VGG19, DenseNet121), and Transformer-based architecture (ViT-S). Figure 4 shows that USEFUL is effective

across different model architectures. Remarkably, when applying to non-CNN architectures, it reduces the test error of SGD to a lower level than that of SAM alone. Detailed results for 3-layer MLP is given in Appendix D.2.

Settings: batch-size, learning rate, and SAM variants. In Appendix D, we confirm the effectiveness of USEFUL for different batch sizes of 128, 256, 512, and different initial learning rates of 0.1, 0.2, 0.4. In Appendix D.4, we also show that USEFUL applied to ASAM (Kwon et al., 2021) can further reduce the test error. ASAM accounts for model parameter re-scaling (Dinh et al., 2017), and uses a scale-invariant sharpness measure with a stronger correlation with the generalization gap, compared to SAM.

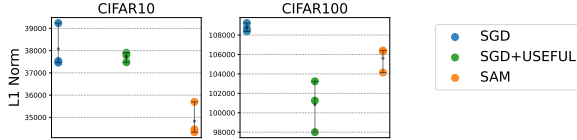


Figure 5: **L1 norm of ResNet18 trained on CIFAR10 and ResNet34 trained on CIFAR100.** Lower L1 norm indicates a sparser solution and stronger implicit regularization properties (Andriushchenko & Flammarion (2022, Section 4.2)). SAM has a lower L1 norm than SGD, and USEFUL further reduces the L1 norm of SGD and SAM.

Table 1: **Sharpness of solution at convergence.** We train ResNet18 on CIFAR10 and measure the maximum Hessian eigenvalue $\lambda_{\max}$ and the bulk spectra measured as $\lambda_{\max}/\lambda_5$.

METRIC	SGD	SGD+USEFUL	SAM
$\lambda_{\max}$	53.8	41.8	12.4
$\lambda_{\max}/\lambda_5$	3.8	1.5	2.4

5.4. USEFUL vs Random Upsampling

Tables 2 and 3 show that USEFUL outperforms SGD and SAM on a randomly upsampled data by up to 0.35% and 1.72% on CIFAR10 and CIFAR100. This clearly confirms that the main benefit of USEFUL is due to the modified distribution. In Appendix D.6, we confirm that upsampling outperforms upweighting for both SAM and SGD.

5.5. USEFUL’s Solution has Similar Properties to SAM

SAM & USEFUL Find Sparser Solutions than SGD. (Andriushchenko & Flammarion, 2022, Section 4.2) showed that SAM’s solution has a better sparsity-inducing property indicated by the L1 norm than the standard ERM. Fig. 5 shows the L1 norm of ResNet18 trained on CIFAR10 and ResNet34 trained on CIFAR100 at the end of training for different methods. We see that USEFUL drives both SGD and SAM to find solutions with smaller L1 norms. This confirms that the solutions found by USEFUL have similar properties to SAM’s solutions.

SAM & USEFUL Find Less Sharp Solutions than SGD.

The maximum Hessian eigenvalue ($\lambda_{\max}$) and the bulk of the spectrum ($\lambda_{\max}/\lambda_5$) (Jastrzebski et al., 2020) are commonly used metrics for sharpness (Keskar et al., 2016; Jastrzebski et al., 2017; Chaudhari et al., 2019; Wen et al., 2019). Table 1 illustrates that training SGD+USEFUL on CIFAR10 reduces sharpness metrics significantly compared to SGD. Interestingly, SGD+USEFUL achieves the smallest bulk of the spectrum among all the methods. This further confirms that the solutions found by USEFUL have similar properties to SAM’s solutions.

SAM & USEFUL Reduce Forgetting Scores. Forgetting scores (Toneva et al., 2018) count the number of times

Table 2: **Test classification errors of SGD** for different data selection strategies. Results are averaged over 3 seeds.

SELECTION STRATEGY	CIFAR10	CIFAR100
ORIGINAL DATA	5.07 ± 0.04	24.06 ± 0.47
RANDOM UPSAMPLING	5.14 ± 0.11	24.30 ± 1.19
USEFUL	4.79 ± 0.05	22.58 ± 0.08

Table 3: **Test classification errors of SAM** for different data selection strategies. Results are averaged over 3 seeds.

SELECTION STRATEGY	CIFAR10	CIFAR100
ORIGINAL DATA	4.23 ± 0.08	21.25 ± 0.11
RANDOM UPSAMPLING	4.17 ± 0.03	21.11 ± 0.04
USEFUL	4.04 ± 0.06	20.66 ± 0.44

an example is misclassified after being correctly classified during training and is an indicator of the learning-difficulty of examples. We show in Appendix D.3 that both SAM and USEFUL successfully reduce the forgetting scores, thus learn difficult features faster than SGD. This aligns with our theoretical analysis in Theorem 3.4 and results on the toy datasets. By upsampling difficult examples in the dataset, they contribute more to learning and hence SGD+USEFUL learns difficult features faster than SGD.

USEFUL also Benefits Distribution Shift. While our main contribution is providing a novel and effective method to improve the in-distribution generalization performance, we conduct experiments confirming the benefits of our method to improving out-of-distribution generalization performance. We discuss this experiment and provide the results in Appendix D.5. On Waterbirds dataset (Sagawa et al., 2019) with strong spurious correlation (95%), both SAM and USEFUL successfully improve the performance on the balanced test set by 6.21% and 5.8%, respectively. We also show the applicability of USEFUL to fine-tuning a ResNet50 pre-trained on ImageNet.

6. Conclusion

In this paper, we made the first attempt to improve the in-distribution generalization performance of machine learning methods by modifying the distribution of training data. We first analyzed learning dynamics of sharpness-aware minimization (SAM), and attributed its superior performance over GD to mitigating the simplicity bias, and learning easy and difficult features more evenly. Inspired by the training dynamics of SAM, we upsampled the examples that do not contain any easy features to alleviate the simplicity bias. This allows learning features more uniformly, thus improving the model performance. Our method boosts the performance of image classification models training with either SGD or SAM and easily stacks with data augmentation.

References

- Abbas, A., Tirumala, K., Simig, D., Ganguli, S., and Morcos, A. S. Semdedup: Data-efficient learning at web-scale through semantic deduplication. *arXiv preprint arXiv:2303.09540*, 2023.
- Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.
- Andriushchenko, M. and Flammarion, N. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pp. 639–668. PMLR, 2022.
- Andriushchenko, M., Bahri, D., Mobahi, H., and Flammarion, N. Sharpness-aware minimization leads to low-rank features. *arXiv preprint arXiv:2305.16292*, 2023.
- Bartlett, P. L., Long, P. M., and Bousquet, O. The dynamics of sharpness-aware minimization: Bouncing across ravines and drifting towards wide minima. *Journal of Machine Learning Research*, 24(316):1–36, 2023.
- Belkin, M., Hsu, D., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.
- Cao, Y., Chen, Z., Belkin, M., and Gu, Q. Benign overfitting in two-layer convolutional neural networks. *Advances in neural information processing systems*, 35:25237–25250, 2022.
- Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y., and Park, S. Swad: Domain generalization by seeking flat minima. *Advances in Neural Information Processing Systems*, 34:22405–22418, 2021.
- Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019.
- Chen, Z., Deng, Y., Wu, Y., Gu, Q., and Li, Y. Towards understanding the mixture-of-experts layer in deep learning. *Advances in neural information processing systems*, 35: 23049–23062, 2022.
- Chen, Z., Zhang, J., Kou, Y., Chen, X., Hsieh, C.-J., and Gu, Q. Why does sharpness-aware minimization generalize better than sgd? *arXiv preprint arXiv:2310.07269*, 2023.
- Coates, A., Ng, A., and Lee, H. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.
- Darlow, L. N., Crowley, E. J., Antoniou, A., and Storkey, A. J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.
- Deng, Y., Yang, Y., Mirzsoleiman, B., and Gu, Q. Robust learning with progressive data expansion against spurious correlation. *arXiv preprint arXiv:2306.04949*, 2023.
- Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017.
- Du, J., Yan, H., Feng, J., Zhou, J. T., Zhen, L., Goh, R. S. M., and Tan, V. Y. Efficient sharpness-aware minimization for improved training of neural networks. *arXiv preprint arXiv:2110.03141*, 2021.
- El Hanchi, A., Stephens, D., and Maddison, C. Stochastic reweighted gradient descent. In *International Conference on Machine Learning*, pp. 8359–8374. PMLR, 2022.
- Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.
- Gadre, S. Y., Ilharco, G., Fang, A., Hayase, J., Smyrnis, G., Nguyen, T., Marten, R., Wortsman, M., Ghosh, D., Zhang, J., et al. Datacomp: In search of the next generation of multimodal datasets. *arXiv preprint arXiv:2304.14108*, 2023.
- Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pp. 2232–2241. PMLR, 2019.
- Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., and Srebro, N. Implicit regularization in matrix factorization. *Advances in neural information processing systems*, 30, 2017.
- Hermann, K. and Lampinen, A. What shapes feature representations? exploring datasets, architectures, and training. *Advances in Neural Information Processing Systems*, 33: 9995–10006, 2020.
- Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

- Hu, W., Xiao, L., Adlam, B., and Pennington, J. The surprising simplicity of the early-time learning dynamics of neural networks. *Advances in Neural Information Processing Systems*, 33:17116–17128, 2020.
- James, G., Witten, D., Hastie, T., Tibshirani, R., et al. *An introduction to statistical learning*, volume 112. Springer, 2013.
- Jastrzebski, S., Szymczak, M., Fort, S., Arpit, D., Tabor, J., Cho, K., and Geras, K. The break-even point on optimization trajectories of deep neural networks. *arXiv preprint arXiv:2002.09572*, 2020.
- Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., and Storkey, A. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.
- Jelassi, S. and Li, Y. Towards understanding how momentum improves generalization in deep learning. In *International Conference on Machine Learning*, pp. 9965–10040. PMLR, 2022.
- Johnson, T. B. and Guestrin, C. Training deep models faster with robust, approximate importance sampling. *Advances in Neural Information Processing Systems*, 31, 2018.
- Kalimeris, D., Kaplun, G., Nakkiran, P., Edelman, B., Yang, T., Barak, B., and Zhang, H. Sgd on neural networks learns functions of increasing complexity. *Advances in neural information processing systems*, 32, 2019.
- Katharopoulos, A. and Fleuret, F. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pp. 2525–2534. PMLR, 2018.
- Kaur, S., Cohen, J., and Lipton, Z. C. On the maximum hessian eigenvalue and generalization. In *Proceedings on*, pp. 51–65. PMLR, 2023.
- Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
- Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- Kirichenko, P., Izmailov, P., and Wilson, A. G. Last layer re-training is sufficient for robustness to spurious correlations. *arXiv preprint arXiv:2204.02937*, 2022.
- Kou, Y., Chen, Z., Chen, Y., and Gu, Q. Benign overfitting in two-layer relu convolutional neural networks. In *International Conference on Machine Learning*, pp. 17615–17659. PMLR, 2023.
- Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.
- Kwon, J., Kim, J., Park, H., and Choi, I. K. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pp. 5905–5914. PMLR, 2021.
- Le, Y. and Yang, X. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.
- Lee, K., Ippolito, D., Nystrom, A., Zhang, C., Eck, D., Callison-Burch, C., and Carlini, N. Deduplicating training data makes language models better. *arXiv preprint arXiv:2107.06499*, 2021.
- Li, J., Socher, R., and Hoi, S. C. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.
- Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in neural information processing systems*, 32, 2019.
- Liu, C., Zoph, B., Neumann, M., Shlens, J., Hua, W., Li, L.-J., Fei-Fei, L., Yuille, A., Huang, J., and Murphy, K. Progressive neural architecture search. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 19–34, 2018.
- Mei, S. and Montanari, A. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.
- Müller, S. G. and Hutter, F. Trivialaugment: Tuning-free yet state-of-the-art data augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 774–782, 2021.
- Nakkiran, P., Kaplun, G., Bansal, Y., Yang, T., Barak, B., and Sutskever, I. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.
- Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. *arXiv preprint arXiv:1412.6614*, 2014.
- Papayan, V. The full spectrum of deepnet Hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

- Pezeshki, M., Kaba, O., Bengio, Y., Courville, A. C., Precup, D., and Lajoie, G. Gradient starvation: A learning proclivity in neural networks. *Advances in Neural Information Processing Systems*, 34:1256–1272, 2021.
- Pham, H., Guan, M., Zoph, B., Le, Q., and Dean, J. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pp. 4095–4104. PMLR, 2018.
- Puli, A. M., Zhang, L., Wald, Y., and Ranganath, R. Don’t blame dataset shift! shortcut learning due to gradients and cross entropy. *Advances in Neural Information Processing Systems*, 36:71874–71910, 2023.
- Puli, A. M., Zhang, L., Wald, Y., and Ranganath, R. Don’t blame dataset shift! shortcut learning due to gradients and cross entropy. *Advances in Neural Information Processing Systems*, 36, 2024.
- Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. The MIT Press, 2009. ISBN 0262170051.
- Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21 (140):1–67, 2020.
- Roux, N., Schmidt, M., and Bach, F. A stochastic gradient method with an exponential convergence rate for finite training sets. *Advances in neural information processing systems*, 25, 2012.
- Sagawa, S., Koh, P. W., Hashimoto, T. B., and Liang, P. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. *arXiv preprint arXiv:1911.08731*, 2019.
- Schmidt, M., Le Roux, N., and Bach, F. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 162:83–112, 2017.
- Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. *Advances in Neural Information Processing Systems*, 33: 9573–9585, 2020.
- Springer, J. M., Nagarajan, V., and Raghunathan, A. Sharpness-aware minimization enhances feature quality via balanced learning. In *The Twelfth International Conference on Learning Representations*, 2024.
- Steinhardt, J., Koh, P. W. W., and Liang, P. S. Certified defenses for data poisoning attacks. *Advances in neural information processing systems*, 30, 2017.
- Toneva, M., Sordoni, A., Combes, R. T. d., Trischler, A., Bengio, Y., and Gordon, G. J. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.
- Valle-Perez, G., Camargo, C. Q., and Louis, A. A. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *International Conference on Learning Representations*, 2018.
- Wang, P., Zhang, Z., Lei, Z., and Zhang, L. Sharpness-aware gradient matching for domain generalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3769–3778, 2023.
- Wen, K., Ma, T., and Li, Z. How does sharpness-aware minimization minimize sharpness? *arXiv preprint arXiv:2211.05729*, 2022.
- Wen, Y., Luk, K., Gazeau, M., Zhang, G., Chan, H., and Ba, J. An empirical study of large-batch stochastic gradient descent with structured covariance noise. *arXiv preprint arXiv:1902.08234*, 2019.
- Yao, Z., Gholami, A., Shen, S., Mustafa, M., Keutzer, K., and Mahoney, M. Adahessian: An adaptive second order optimizer for machine learning. In *proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 10665–10673, 2021.
- Yuan, L., Chen, Y., Wang, T., Yu, W., Shi, Y., Jiang, Z.-H., Tay, F. E., Feng, J., and Yan, S. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 558–567, 2021.
- Zhai, R., Dan, C., Kolter, Z., and Ravikumar, P. Understanding why generalized reweighting does not improve over erm. *arXiv preprint arXiv:2201.12293*, 2022.
- Zhao, P. and Zhang, T. Stochastic optimization with importance sampling for regularized loss minimization. In *international conference on machine learning*, pp. 1–9. PMLR, 2015.
- Zheng, Y., Zhang, R., and Mao, Y. Regularizing neural networks via adversarial model perturbation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8156–8165, 2021.
- Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A. Formal Proofs

A.1. Proof of Theorem 3.2

Notation. In this paper, we use lowercase letters, lowercase boldface letters, and uppercase boldface letters to respectively denote scalars (a), vectors ($\mathbf{v}$), and matrices ($\mathbf{W}$). For a vector $\mathbf{v}$, we use $\|\mathbf{v}\|_2$ to denote its Euclidean norm. Given two sequence $\{x_n\}$ and $\{y_n\}$, we denote $x_n = O(y_n)$ if $|x_n| \leq C_1|y_n|$ for some absolute positive constant C_1 , $x_n = \Omega(y_n)$ if $|x_n| \geq C_2|y_n|$ for some absolute positive constant C_2 , and $x_n = \Theta(y_n)$ if $C_3|y_n| \leq |x_n| \leq C_4|y_n|$ for some absolute constant $C_3, C_4 > 0$. In addition, we use $\tilde{O}(\cdot)$, $\tilde{\Omega}(\cdot)$, and $\tilde{\Theta}(\cdot)$ to hide logarithmic factors in these notations. Furthermore, we denote $x_n = \text{poly}(y_n)$ if $x_n = O(y_n^D)$ for some positive constant D , and $x_n = \text{polylog}(y_n)$ if $x_n = \text{poly}(\log(y_n))$.

First, we have the following assumption for the model weight initialization.

Assumption A.1 (Weight initialization). Assume that we initialize $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ such that for all $j \in [J]$, $\langle \mathbf{w}_j^{(0)}, \mathbf{v}_e \rangle, \langle \mathbf{w}_j^{(0)}, \mathbf{v}_d \rangle \geq \rho > 0$.

The above assumption is reasonable because we later show that both sequences $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle$ and $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle$ are non-decreasing. So, we can obtain the above initialization by training the model for several iterations. For simplicity of the notation, we assume that αN is an integer and the first αN data examples have the easy feature while the rest do not. Before going into the analysis, we denote the derivative of a data example i at iteration t to be

$$l_i^{(t)} = \frac{\exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)}))}{1 + \exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)}))} = \text{sigmoid}(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)})). \quad (13)$$

Lemma A.2 (Gradient). Let the loss function $\mathcal{L}$ be as defined in Equation 2. For $t \geq 0$ and $j \in [J]$, the gradient of the loss $\mathcal{L}(\mathbf{W}^{(t)})$ with regard to neuron $\mathbf{w}_j^{(t)}$ is

$$\begin{aligned} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) &= -\frac{3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right). \end{aligned} \quad (14)$$

Proof. We have the following gradient

$$\begin{aligned} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) &= -\frac{1}{N} \sum_{i=1}^N \frac{\exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)}))}{1 + \exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)}))} \cdot y_i f'(\mathbf{x}_i; \mathbf{W}^{(t)}) \\ &= -\frac{3}{N} \sum_{i=1}^N l_i^{(t)} y_i \sum_{p=1}^P \langle \mathbf{w}_j^{(t)}, \mathbf{x}^{(p)} \rangle^2 \cdot \mathbf{x}^{(p)} \\ &= -\frac{3}{N} \sum_{i=1}^N l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) \end{aligned}$$

□

With the above formula of gradient, we have the following equations:

Easy feature gradient. The projection of the gradient on $\mathbf{v}_e$ is

$$\langle \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_e \rangle = -\frac{3\beta_e^3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \quad (15)$$

Difficult feature gradient. The projection of the gradient on $\mathbf{v}_d$ is

$$\langle \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_d \rangle = -\frac{3\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \quad (16)$$

Noise gradient. The projection of the gradient on ξ_i is

$$\langle \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \xi_i \rangle = -\frac{3}{N} \left(l_i^{(t)} y_i \langle \mathbf{w}_j^{(t)}, \xi_i \rangle^2 \|\xi_i\|_2^2 + \sum_{k=1, k \neq i}^N l_k^{(t)} y_k \langle \mathbf{w}_j^{(t)}, \xi_k \rangle^2 \langle \xi_k, \xi_i \rangle \right) \quad (17)$$

Derivative of data example i . For $1 \leq i \leq \alpha N$, $l_i^{(t)}$ can be rewritten as

$$l_i^{(t)} = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 - y_i \langle \mathbf{w}_j^{(t)}, \xi_i \rangle^3 \right) \quad (18)$$

while for $\alpha N + 1 \leq i \leq N$, $l_i^{(t)}$ can be rewritten as

$$l_i^{(t)} = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - y_i \langle \mathbf{w}_j^{(t)}, \xi_i \rangle^3 \right) \geq \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 - y_i \langle \mathbf{w}_j^{(t)}, \xi_i \rangle^3 \right) \quad (19)$$

Note that $0 < l_i^{(t)} < 1$ due to the property of the sigmoid function. Furthermore, we similarly consider that the sum of the sigmoid terms for all time steps is bounded up to a logarithmic dependence (Chen et al., 2022). The sigmoid term is considered small for a κ such that

$$\sum_{t=0}^T \frac{1}{1 + \exp(\kappa)} \leq \tilde{O}(1), \quad (20)$$

which implies $\kappa \geq \tilde{\Omega}(1)$.

We present the detailed proofs that build up to Theorem A.8. We begin by considering the update for the easy and difficult features.

Lemma A.3 (Easy feature update.). *For all $t \geq 0$ and $j \in [J]$, the easy feature update is*

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_1(t) \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2, \quad (21)$$

where $g_1(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 \right)$.

Proof. Plugging the update rule of GD, we have

$$\begin{aligned} \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \eta \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_e \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \frac{3\eta\beta_e^3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_1(t) \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2, \end{aligned}$$

where the last equality holds due to Lemma B.5. □

Similarly, we obtain the following update rule for difficult features.

Lemma A.4 (Difficult feature update.). *For all $t \geq 0$ and $j \in [J]$, the easy feature update is*

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \frac{3\eta\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2, \quad (22)$$

which gives

$$\tilde{\Theta}(\eta)\beta_d^3 g_1(t) \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \leq \tilde{\Theta}(\eta)\beta_d^3 (\alpha g_1(t) + 1 - \alpha) \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \quad (23)$$

where $g_1(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 \right)$.

Proof. Plugging the update rule of GD, we have

$$\begin{aligned} \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle &= \langle \mathbf{w}_j^{(t)} - \eta \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_d \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \frac{3\eta\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \end{aligned}$$

From Lemma B.5, we have for $1 \leq i \leq \alpha N$, $l_i^{(t)} = \Theta(1)g_1(t)$ and for $\alpha N + 1 \leq i \leq N$, $\Theta(1)g_1(t) \leq l_i^{(t)} \leq 1$. Combining with the above equality, we obtain the desired inequalities. $\square$

Next, we simplify the two above update rules in the early training stage.

Lemma A.5 (Easy feature update in early iterations). *Let $T_0 > 0$ be such that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$. For $t \in [0, T_0]$, the easy feature update has the following rule*

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta)\alpha\beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2, \quad (24)$$

Proof. Let $T_0 > 0$ be such that either $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$ or $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle \geq \tilde{\Omega}(1/\beta_d)$. We will show later that the first condition will be met and we have $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle \leq \tilde{\Omega}(1/\beta_d)$ for all $j \in [J]$ and $t \in [0, T_0]$.

Recall that $g_1(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 \right)$. Then, for $t \in [0, T_0]$, we have

$$\begin{aligned} g_1(t) &= \frac{1}{1 + \exp(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3)} \\ &\geq \frac{1}{1 + \exp(\kappa + \kappa)} \\ &= \frac{1}{1 + \exp(\tilde{\Omega}(1))}, \end{aligned}$$

where the first inequality holds due to $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle \leq \kappa/(J^{1/3}\beta_e)$ and $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \leq \kappa/(J^{1/3}\beta_d)$ for $t \in [0, T_0]$ (Deng et al., 2023)[Lemma E.3]. Therefore, similar to (Deng et al., 2023; Jelassi & Li, 2022), we have $g_1(t) = \Theta(1)$ in the early iterations. This implies the result in Lemma A.3 as

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta)\alpha\beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2. \quad (25)$$

$\square$

Similarly, we obtain the following simplified update rule for difficult features in the early iterations.

Lemma A.6 (Difficult feature update in early iterations). *Let $T_0 > 0$ be such that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$. For $t \in [0, T_0]$, the easy feature update has the following rule*

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta)\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2, \quad (26)$$

We next show that GD will learn the easy feature quicker than learning the difficult feature.

Lemma A.7. Assume $\eta = \tilde{O}(\beta_d \sigma_0)$. Let T_0 be the iteration number that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle$ reaches $\tilde{\Omega}(1/\beta_e) = \Theta(1)$. Then, we have for all $t \leq T_0$, it holds that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.

Proof. Among all the possible indices $j \in [J]$, we focus on the index $j^* = \arg \max_{j \in [J]} \langle \mathbf{w}_j^{(0)}, \mathbf{v}_e \rangle$. Therefore, for $C_t = \alpha \beta_e^3 = \Theta(1)$, we apply Lemma B.2 with two positive sequences $\langle \mathbf{w}_{j^*}^{(t)}, \mathbf{v}_e \rangle$ and $\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle$ defined in Lemmas A.5 and A.6 and get

$$\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \leq O(\langle \mathbf{w}_{j^*}^{(0)}, \mathbf{v}_d \rangle) = \tilde{O}(\sigma_0) \quad (27)$$

for all $j \in [J]$. $\square$

Theorem A.8 (Restatement of Theorem 3.2). We consider training a two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ on the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ with $\alpha^{1/3} \beta_e > \beta_d$. After training with GD in Equation 3 for T_0 iterations where

$$T_0 = \frac{\tilde{\Theta}(1)}{\eta \alpha \beta_e^3 \sigma_0} + \tilde{\Theta}(1) \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil, \quad (28)$$

for all $j \in [J]$ and $t \in [0, T_0)$, we have

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2. \quad (29)$$

$$\langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \quad (30)$$

After training for T_0 iterations, with high probability, the learned weight has the following properties: (1) it learns the easy feature $\mathbf{v}_e$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$; (2) it does not learn the difficult feature $\mathbf{v}_d$: $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.

Proof. From the results of Lemmas A.5- A.7, it remains to calculate the time T_0 . Plugging $v = \tilde{\Omega}(1/\beta_e)$, $m = M = \tilde{\Theta}(\eta) \alpha \beta_e^3$, $z_0 = \tilde{O}(\sigma_0)$ into Lemma B.3, we have T_0 as

$$T_0 = \frac{\tilde{\Theta}(1)}{\eta \alpha \beta_e^3 \sigma_0} + \tilde{\Theta}(1) \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil \quad (31)$$

$\square$

A.2. Proof of Theorem 3.3

Before going into the analysis, we denote the derivative of a data example i at iteration t to be

$$l_{i, \epsilon}^{(t)} = \frac{\exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)} + \epsilon^{(t)}))}{1 + \exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)} + \epsilon^{(t)}))} = \text{sigmoid}(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)} + \epsilon^{(t)})), \quad (32)$$

where $\epsilon^{(t)} = \rho^{(t)} \nabla \mathcal{L}(\mathbf{W}^{(t)})$ is the weighted ascent direction at the current parameter $\mathbf{W}^{(t)}$. We denote the weight vector of the j -th filter after being perturbed by SAM as

$$\mathbf{w}_{j, \epsilon}^{(t)} = \mathbf{w}_j^{(t)} + \epsilon_j^{(t)} = \mathbf{w}_j^{(t)} + \rho^{(t)} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \quad (33)$$

where $\rho^{(t)} = \rho / \|\nabla \mathcal{L}(\mathbf{W}^{(t)})\|_F$.

First, we have the following inequalities regarding the gradient norm:

$$\|\nabla \mathcal{L}(\mathbf{W}^{(t)})\|_F \geq \|\nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)})\| \quad (34)$$

$$= \langle \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) \rangle^{1/2} \quad (35)$$

$$= \left[\left(\frac{3\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \right)^2 + \left(\frac{3\beta_e^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \right)^2 + \left\| \frac{3}{N} \sum_{i=1}^N l_i^{(t)} y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right\|^2 \right]^{1/2} \quad (36)$$

Thus,

$$\left\| \nabla \mathcal{L}(\mathbf{W}^{(t)}) \right\|_F \geq \frac{3\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \quad (37)$$

$$\left\| \nabla \mathcal{L}(\mathbf{W}^{(t)}) \right\|_F \geq \frac{3\beta_e^3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \quad (38)$$

Lemma A.9 (Gradient). *Let the loss function $\mathcal{L}$ be as defined in Equation 2. For $t \geq 0$ and $j \in [J]$, the gradient of the loss $\mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)})$ with regard to neuron $\mathbf{w}_{j,\epsilon}^{(t)}$ is*

$$\begin{aligned} \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) &= -\frac{3}{N} \sum_{i=1}^{\alpha N} l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) \end{aligned} \quad (39)$$

Proof. We have the following gradient

$$\begin{aligned} \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) &= -\frac{1}{N} \sum_{i=1}^N \frac{\exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}))}{1 + \exp(-y_i f(\mathbf{x}_i; \mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}))} \cdot y_i f'(\mathbf{x}_i; \mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) \\ &= -\frac{3}{N} \sum_{i=1}^N l_{i,\epsilon}^{(t)} y_i \sum_{p=1}^P \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{x}^{(p)} \rangle^2 \cdot \mathbf{x}^{(p)} \\ &= -\frac{3}{N} \sum_{i=1}^{\alpha N} l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) \end{aligned}$$

□

With the above formula of gradient, we have the projection of perturbed weight on $\mathbf{v}_e$ is

$$\begin{aligned} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \langle \boldsymbol{\epsilon}_j^{(t)}, \mathbf{v}_e \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \langle \rho^{(t)} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_e \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \frac{3\rho^{(t)}\beta_e^3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \end{aligned} \quad (40)$$

From Equations 38 and 40, we have

$$0 \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \rho \leq \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle \quad (41)$$

Similarly, the projection of perturbed weight on $\mathbf{v}_d$ is

$$\langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \frac{3\rho^{(t)}\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \quad (42)$$

$$0 \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \rho \leq \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \quad (43)$$

Easy feature gradient. The projection of the gradient on $\mathbf{v}_e$ is

$$\langle \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}), \mathbf{v}_e \rangle = -\frac{3\beta_e^3}{N} \sum_{i=1}^{\alpha N} l_{i,\epsilon}^{(t)} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \quad (44)$$

Difficult feature gradient. The projection of the gradient on $\mathbf{v}_d$ is

$$\langle \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}), \mathbf{v}_d \rangle = -\frac{3\beta_d^3}{N} \sum_{i=1}^N l_{i,\epsilon}^{(t)} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \quad (45)$$

Noise gradient. The projection of the gradient on $\boldsymbol{\xi}_i$ is

$$\langle \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}), \boldsymbol{\xi}_i \rangle = -\frac{3}{N} \left(l_{i,\epsilon}^{(t)} y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \|\boldsymbol{\xi}_i\|_2^2 + \sum_{k=1, k \neq i}^N l_{k,\epsilon}^{(t)} y_k \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_k \rangle^2 \langle \boldsymbol{\xi}_k, \boldsymbol{\xi}_i \rangle \right) \quad (46)$$

Derivative of data example i . For $1 \leq i \leq \alpha N$, $l_{i,\epsilon}^{(t)}$ can be rewritten as

$$l_{i,\epsilon}^{(t)} = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3 - y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^3 \right) \quad (47)$$

while for $\alpha N + 1 \leq i \leq N$, $l_i^{(t)}$ can be rewritten as

$$l_{i,\epsilon}^{(t)} = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^3 \right) \geq \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3 - y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^3 \right) \quad (48)$$

We present the detailed proofs that build up to Theorem A.15. We begin by considering the update for the easy and difficult features.

Lemma A.10 (Easy feature update.). *For all $t \geq 0$ and $j \in [J]$, the easy feature update is*

$$\begin{aligned} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_2(t) (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \rho)^2 &\leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_2(t) \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_2(t) (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle)^2 \end{aligned} \quad (49)$$

where $g_2(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3 \right)$.

Proof. Plugging the update rule of SAM, we have

$$\begin{aligned} \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \eta \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}), \mathbf{v}_e \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \frac{3\eta \alpha \beta_e^3}{N} \sum_{i=1}^N l_{i,\epsilon}^{(t)} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 g_2(t) \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2, \end{aligned}$$

where the last equality holds due to Lemma B.10. Combining with Equation 41, we obtain the desired inequalities. $\square$

Similarly, we obtain the following update rule for difficult features.

Lemma A.11 (Difficult feature update.). *For all $t \geq 0$ and $j \in [J]$, the difficult feature update is*

$$\begin{aligned} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 g_2(t) (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \rho)^2 &\leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \frac{3\eta\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 (\alpha g_2(t) + 1 - \alpha) (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle)^2 \end{aligned} \quad (50)$$

where $g_2(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3 \right)$.

Proof. Plugging the update rule of GD, we have

$$\begin{aligned} \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle &= \langle \mathbf{w}_j^{(t)} - \eta \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}), \mathbf{v}_d \rangle \\ &= \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \frac{3\eta\beta_d^3}{N} \sum_{i=1}^N l_i^{(t)} \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \end{aligned}$$

From Lemma B.5, we have for $1 \leq i \leq \alpha N$, $l_i^{(t)} = \Theta(1)g_1(t)$ and for $\alpha N + 1 \leq i \leq N$, $\Theta(1)g_1(t) \leq l_i^{(t)} \leq 1$. Combining with the above equality and Equation 43, we obtain the desired inequalities. $\square$

Next, we simplify the two above update rules in the early training stage.

Lemma A.12 (Easy feature update in early iterations). *Let $T_0 > 0$ be such that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$. For $t \in [0, T_0]$, the easy feature update has the following rule*

$$\tilde{\Theta}(\eta) \alpha \beta_e^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \rho)^2 \leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle)^2 \quad (51)$$

Proof. Let $T_0 > 0$ be such that either $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$ or $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle \geq \tilde{\Omega}(1/\beta_d)$. We will show later that the first condition will be met and we have $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle \leq \tilde{\Omega}(1/\beta_d)$ for all $j \in [J]$ and $t \in [0, T_0]$.

Recall that $g_2(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3 \right)$. Then, for $t \in [0, T_0]$, we have

$$\begin{aligned} g_2(t) &= \frac{1}{1 + \exp(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3)} \\ &\geq \frac{1}{1 + \exp(\kappa + \kappa)} \\ &= \frac{1}{1 + \exp(\tilde{\Omega}(1))}, \end{aligned}$$

where the first inequality holds due to $\langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle \leq \kappa/(J^{1/3}\beta_e)$ and $\langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle \leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \leq \kappa/(J^{1/3}\beta_d)$ for $t \in [0, T_0]$. Therefore, we have $g_2(t) = \Theta(1)$ in the early iterations. Replacing $g_2(t) = \Theta(1)$ into the results of Lemma A.10, we obtain the desired results. $\square$

Similarly, we obtain the following simplified update rule for difficult features in the early iterations.

Lemma A.13 (Difficult feature update in early iterations). *Let $T_0 > 0$ be such that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$. For $t \in [0, T_0]$, the easy feature update has the following rule*

$$\tilde{\Theta}(\eta) \beta_d^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \rho)^2 \leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle - \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle \leq \tilde{\Theta}(\eta) \beta_d^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle)^2 \quad (52)$$

We next show that SAM will learn the easy feature quicker than the difficult one.

Lemma A.14. *Assume $\eta = \tilde{o}(\beta_d \sigma_0)$. Let T_0 be the iteration number that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle$ reaches $\tilde{\Omega}(1/\beta_e) = \Theta(1)$. Then, we have for all $t \leq T_0$, it holds that $\max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.*

Proof. Among all the possible indices $j \in [J]$, we focus on the index $j^* = \arg \max_{j \in [J]} \langle \mathbf{w}_j^{(0)}, \mathbf{v}_e \rangle$. Therefore, for $C_t = \alpha \beta_e^3 = \Theta(1)$, we apply Lemma B.2 with two positive sequences $\langle \mathbf{w}_{j^*}^{(t)}, \mathbf{v}_e \rangle$ and $\langle \mathbf{w}_{j^*}^{(t)}, \mathbf{v}_d \rangle$ defined in Lemmas A.12 and A.13 and get

$$\langle \mathbf{w}_{j^*}^{(t)}, \mathbf{v}_d \rangle \leq O(\langle \mathbf{w}_{j^*}^{(0)}, \mathbf{v}_d \rangle) = \tilde{O}(\sigma_0) \quad (53)$$

for all $j \in [J]$. $\square$

Theorem A.15 (Restatement of Theorem 3.3). *We consider training a two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$ on the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ with $\alpha^{1/3} \beta_e > \beta_d$. After training with SAM in Equation 3 for T_0 iterations where*

$$T_0 = \frac{\tilde{\Theta}(\sigma_0)}{\eta \alpha \beta_e^3 (\sigma_0 - \rho)^2} + \frac{\tilde{\Theta}(\sigma_0^2)}{(\sigma_0 - \rho)^2} \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil, \quad (54)$$

for all $j \in [J]$ and $t \in [0, T_0)$, we have

$$\begin{aligned} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle - \rho)^2 &\leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_e \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle + \tilde{\Theta}(\eta) \alpha \beta_e^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle)^2 \end{aligned} \quad (55)$$

$$\begin{aligned} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle - \rho)^2 &\leq \langle \mathbf{w}_j^{(t+1)}, \mathbf{v}_d \rangle = \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \\ &\leq \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle + \tilde{\Theta}(\eta) \beta_d^3 (\langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle)^2 \end{aligned} \quad (56)$$

After training for T_0 iterations, with high probability, the learned weight has the following properties: (1) it learns the easy feature $\mathbf{v}_e : \max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_e \rangle \geq \tilde{\Omega}(1/\beta_e)$; (2) it does not learn the difficult feature $\mathbf{v}_d : \max_{j \in [J]} \langle \mathbf{w}_j^{(T_0)}, \mathbf{v}_d \rangle = \tilde{O}(\sigma_0)$.

Proof. With the results of Lemmas A.12- A.14, it remains to calculate the time T_0 . Plugging $v = \tilde{\Omega}(1/\beta_e)$, $m = M = \tilde{\Theta}(\eta) \alpha \beta_e^3$, $z_0 = \tilde{O}(\sigma_0)$ into Lemma B.3, we have T_0 as

$$T_0 = \frac{\tilde{\Theta}(\sigma_0)}{\eta \alpha \beta_e^3 (\sigma_0 - \rho)^2} + \frac{\tilde{\Theta}(\sigma_0^2)}{(\sigma_0 - \rho)^2} \left\lceil \frac{-\log(\sigma_0 \beta_e)}{\log(2)} \right\rceil \quad (57)$$

$\square$

A.3. Proof of Theorem 3.4

In this section, we show that SAM learns easy and difficult features at a more uniform speed. To ease the notation, we denote $G_e^{(t)} = \max_{j \in [J]} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle$ and $G_d^{(t)} = \max_{j \in [J]} \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle$ for model weights trained with GD. Similarly, we denote $S_e^{(t)}$ and $S_d^{(t)}$ for model weights trained with SAM. We use $\hat{S}_e^{(t)}$ and $\hat{S}_d^{(t)}$ to denote the inner products with perturbed weights. We simplify Equation 40 and 42 for early iterations $t \leq T_0$ as

$$\hat{S}_e^{(t)} = S_e^{(t)} - \tilde{\Theta}(1) \rho^{(t)} \alpha \beta_e^3 (S_e^{(t)})^2 \quad (58)$$

$$\hat{S}_d^{(t)} = S_d^{(t)} - \tilde{\Theta}(1) \rho^{(t)} \beta_d^3 (S_d^{(t)})^2 \quad (59)$$

Before introducing the theorem, we assume that the model is initialized in favor of the easy feature, i.e. $G_e^{(0)} - G_d^{(0)} \geq \rho$. This is reasonable as a consequence of Theorem A.8 because we can just train the model for several iterations to achieve this initialization (similar argument for Assumption A.1).

Theorem A.16 (Restatement of Theorem 3.4). *Consider the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ using the two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. Assume that the easy feature strength is significantly larger $\alpha^{1/3} \beta_e > \beta_d$. Training the same model initialization, we have that for every*

iteration $t \leq T_0$

$$\rho + S_d^{(t)} \leq S_e^{(t)} \quad (60)$$

$$\hat{S}_d^{(t)} < \hat{S}_e^{(t)} \quad (61)$$

$$S_e^{(t)} \leq G_e^{(t)} \quad (62)$$

$$S_d^{(t)} \leq G_d^{(t)} \quad (63)$$

$$S_e^{(t)} - S_d^{(t)} \leq G_e^{(t)} - G_d^{(t)} \quad (64)$$

Proof. We prove this by induction. For $t = 0$, the above hypotheses immediately hold because we use train two methods from the same initialization. Particularly, we have $0 < S_d^{(0)} = G_d^{(0)} < G_e^{(0)} = S_e^{(0)}$ and $\hat{S}_e^{(0)} - \hat{S}_d^{(0)} \geq S_e^{(0)} - \rho - S_d^{(0)} \geq (G_e^{(0)} - G_d^{(0)}) - \rho \geq 0$.

Assume that the induction hypotheses hold for t , i.e.

$$\rho + S_d^{(t)} \leq S_e^{(t)} \quad (65)$$

$$\hat{S}_d^{(t)} < \hat{S}_e^{(t)} \quad (66)$$

$$S_e^{(t)} \leq G_e^{(t)} \quad (67)$$

$$S_d^{(t)} \leq G_d^{(t)} \quad (68)$$

$$S_e^{(t)} - S_d^{(t)} \leq G_e^{(t)} - G_d^{(t)} \quad (69)$$

We need to prove that they also hold for $t + 1$. From Lemma A.15 and the first two induction hypotheses,

$$\rho + S_d^{(t+1)} = \rho + S_d^{(t)} + \tilde{\Theta}(\eta)\beta_d^3(\hat{S}_d^{(t)})^2 < S_e^{(t)} + \tilde{\Theta}(\eta)\alpha\beta_e^3(\hat{S}_e^{(t)})^2 = S_e^{(t+1)} \quad (70)$$

Then, $\hat{S}_e^{(t+1)} - \hat{S}_d^{(t+1)} \geq S_e^{(t+1)} - \rho - S_d^{(t+1)} \geq 0$. From Equation 41 and Lemma A.12,

$$S_e^{(t+1)} \leq S_e^{(t)} + \tilde{\Theta}(\eta)\alpha\beta_e^3(S_e^{(t)})^2 \leq G_e^{(t)} + \tilde{\Theta}(\eta)\alpha\beta_e^3(G_e^{(t)})^2 \leq G_e^{(t+1)}. \quad (71)$$

Similarly, we have $S_d^{(t+1)} \leq G_d^{(t+1)}$. From Equations 58 and 59,

$$S_d^{(t)} - \hat{S}_d^{(t)} = \tilde{\Theta}(1)\rho^{(t)}\beta_d^3(S_d^{(t)})^2 < \tilde{\Theta}(1)\rho^{(t)}\alpha\beta_e^3(S_e^{(t)})^2 = S_e^{(t)} - \hat{S}_e^{(t)} \quad (72)$$

$$0 \leq \hat{S}_e^{(t)} - \hat{S}_d^{(t)} < S_e^{(t)} - S_d^{(t)} \leq G_e^{(t)} - G_d^{(t)} \quad (73)$$

Combining with Equations 41 and 43, we have

$$(\hat{S}_e^{(t)})^2 - (\hat{S}_d^{(t)})^2 < (S_e^{(t)})^2 - (S_d^{(t)})^2 \leq (G_e^{(t)})^2 - (G_d^{(t)})^2 \quad (74)$$

$$(G_d^{(t)})^2 - (\hat{S}_d^{(t)})^2 < (G_e^{(t)})^2 - (\hat{S}_e^{(t)})^2 \quad (75)$$

$$\tilde{\Theta}(\eta)\beta_d^3((G_d^{(t)})^2 - (\hat{S}_d^{(t)})^2) < \tilde{\Theta}(\eta)\alpha\beta_e^3((G_e^{(t)})^2 - (\hat{S}_e^{(t)})^2) \quad (76)$$

$$G_d^{(t)} - S_d^{(t)} + \tilde{\Theta}(\eta)\beta_d^3((G_d^{(t)})^2 - (\hat{S}_d^{(t)})^2) < G_e^{(t)} - S_e^{(t)} + \tilde{\Theta}(\eta)\alpha\beta_e^3((G_e^{(t)})^2 - (\hat{S}_e^{(t)})^2) \quad (77)$$

$$G_d^{(t+1)} - S_d^{(t+1)} < G_e^{(t+1)} - S_e^{(t+1)} \quad (78)$$

$$S_e^{(t+1)} - S_d^{(t+1)} < G_e^{(t+1)} - G_d^{(t+1)} \quad (79)$$

Therefore, the induction hypotheses hold for $t + 1$. $\square$

A.4. Proof of Theorem 3.5

From Theorem A.16, we have the following result for switching between SAM and GD during training.

Lemma A.17. Consider the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ using the two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. Assume that the noise is sufficiently small (ref. Lemmas B.4 and B.9) and the easy feature strength is significantly larger $\alpha^{1/3}\beta_e > \beta_d$. From any iteration t during early training, the normalized gradient of the one-step SAM update has a larger weight on the difficult feature compared to that of GD.

Proof. First, recall the gradients of GD and SAM are as follows.

$$\begin{aligned} \nabla_{\mathbf{w}_j^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) &= -\frac{3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_i^{(t)} \left(\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) \end{aligned} \quad (80)$$

$$\begin{aligned} \nabla_{\mathbf{w}_{j,\epsilon}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) &= -\frac{3}{N} \sum_{i=1}^{\alpha N} l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^2 \mathbf{v}_e + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) - \\ &\quad \frac{3}{N} \sum_{i=\alpha N+1}^N l_{i,\epsilon}^{(t)} \left(\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^2 \mathbf{v}_d + y_i \langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle^2 \boldsymbol{\xi}_i \right) \end{aligned} \quad (81)$$

Because the noise is sufficiently small, the above equations can be simplified as

$$\nabla_{\mathbf{w}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) = - \left(\frac{3}{N} \sum_{i=1}^N l_i^{(t)} \beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2 \right) \mathbf{v}_d - \left(\frac{3}{N} \sum_{i=1}^{\alpha N} l_i^{(t)} \beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2 \right) \mathbf{v}_e \quad (82)$$

$$\nabla_{\mathbf{w}_\epsilon^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) = - \left(\frac{3}{N} \sum_{i=1}^N l_{i,\epsilon}^{(t)} \beta_d^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle^2 \right) \mathbf{v}_d - \left(\frac{3}{N} \sum_{i=1}^{\alpha N} l_{i,\epsilon}^{(t)} \beta_e^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle^2 \right) \mathbf{v}_e \quad (83)$$

Note that in early training, we have an approximation for the logit terms as $l_i^{(t)} = l_{i,\epsilon}^{(t)} = \Theta(1)$, we can further simplify the gradients as

$$\nabla_{\mathbf{w}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) = - \left(3\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2 \right) \mathbf{v}_d - \left(3\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2 \right) \mathbf{v}_e \quad (84)$$

$$\nabla_{\mathbf{w}_\epsilon^{(t)}} \mathcal{L}(\mathbf{W}^{(t)} + \boldsymbol{\epsilon}^{(t)}) = - \left(3\beta_d^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle^2 \right) \mathbf{v}_d - \left(3\alpha\beta_e^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle^2 \right) \mathbf{v}_e \quad (85)$$

Both gradients of GD and SAM can be decomposed into the linear combination of easy and difficult features. To prove that the normalized gradient of SAM favors the difficult feature compared to GD, it is sufficient to show the ratio of coefficients in SAM is larger than GD. In other words, we need to verify that

$$\frac{3\beta_d^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle^2}{3\alpha\beta_e^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle^2} \geq \frac{3\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{3\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} \quad (86)$$

$$\frac{\langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle} \geq \frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \quad (87)$$

$$\frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle - 3\rho^{(t)} \beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle - 3\rho^{(t)} \alpha \beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} \geq \frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \quad (88)$$

$$1 - 3\rho^{(t)} \beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle \geq 1 - 3\rho^{(t)} \alpha \beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle \quad (89)$$

$$\alpha \beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle \geq \beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle \quad (90)$$

The last inequality holds due to $\alpha\beta_e^3 > \beta_d^3$ and $\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle \geq \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle$ from Theorem A.16. $\square$

From Equation 86 in the above proof, it can be seen clearly that amplifying the difficult feature strength in either GD or SAM, i.e., increasing β_d , places a larger weight on the difficult feature. Thus, we have the next theorem.

Theorem A.18 (Restatement of Theorem 3.5). *Consider the training dataset $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$ that follows the data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$ using the two-layer nonlinear CNN model initialized with $\mathbf{W}^{(0)} \sim \mathcal{N}(0, \sigma_0^2)$. Assume that the noise is sufficiently small (ref. Lemmas B.4 and B.9) and the easy feature strength is significantly larger $\alpha^{1/3}\beta_e > \beta_d$. We have the following results for one-step upsampling, i.e. increasing β_d , from any iteration t during early training*

1. *The normalized gradient of the one-step SAM update has a larger weight on the difficult feature compared to that of GD.*
2. *Amplifying the difficult feature strength puts a larger weight on the difficult feature in the normalized gradients of GD and SAM.*
3. *There exists an upsampling factor k such that the normalized gradient of the one-step GD update on $\mathcal{D}(\beta_e, k\beta_d, \alpha)$ recovers the normalized gradient of the one-step SAM update on $\mathcal{D}(\beta_e, \beta_d, \alpha)$.*

Proof. The first result has already been proved in Lemma A.17. Now, consider increasing the difficult feature strength from β_d to β'_d . Similar to the proof of Corollary A.17, to verify that the normalized of the new normalized gradient of GD favors the difficult feature, it is sufficient to show

$$\frac{(\beta'_d)^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} \geq \frac{\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} \quad (91)$$

which is trivial because $\beta'_d > \beta_d$. Similarly, we can verify the result for SAM. Now, let's find the new coefficient $\beta'_d = k\beta_d (k > 1)$ such that training one-step GD on $\mathcal{D}(\beta_e, \beta'_d, \alpha)$ can recover the normalized gradient of the one-step SAM update on the original data distribution $\mathcal{D}(\beta_e, \beta_d, \alpha)$. Using Equation 86, we have

$$\frac{\beta_d^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle^2}{\beta_e^3 \langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle^2} = \frac{(\beta'_d)^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} \quad (92)$$

$$\frac{\langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}_\epsilon^{(t)}, \mathbf{v}_e \rangle} = k^{3/2} \frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \quad (93)$$

$$\frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle - 3\rho^{(t)}\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle^2}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle - 3\rho^{(t)}\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle^2} = k^{3/2} \frac{\langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{\langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \quad (94)$$

$$k^{3/2} = \frac{1 - 3\rho^{(t)}\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{1 - 3\rho^{(t)}\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \quad (95)$$

$$k = \left(\frac{1 - 3\rho^{(t)}\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{1 - 3\rho^{(t)}\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \right)^{2/3}. \quad (96)$$

Therefore, with $\beta'_d = \left(\frac{1 - 3\rho^{(t)}\beta_d^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_d \rangle}{1 - 3\rho^{(t)}\alpha\beta_e^3 \langle \mathbf{w}^{(t)}, \mathbf{v}_e \rangle} \right)^{2/3} \beta_d$, the normalized gradient of the one-step GD update on $\mathcal{D}(\beta_e, \beta'_d, \alpha)$ is similar to that of the one-step SAM update on $\mathcal{D}(\beta_e, \beta_d, \alpha)$. $\square$

B. Auxiliary Lemmas

Lemma B.1 (Claim D.20, (Allen-Zhu & Li, 2020)). *Considering an increasing sequence $x_t \geq 0$ defined as $x_{t+1} = x_t + \eta C_t (x_t - \rho)^2$ for some $C_t = \Theta(1)$, $0 \leq \rho \leq x_0$, then we have for every $A > x_0$, every $\delta \in (0, 1)$, and every $\eta \in (0, 1]$:*

$$\sum_{t \geq 0, x_t \leq A} \eta C_t \leq \frac{1 + \delta}{x_0} + \frac{O(\eta(A - \rho)^2) \log(A/x_0)}{x_0^2 \log(1 + \delta)} \quad (97)$$

$$\sum_{t \geq 0, x_t \leq A} \eta C_t \geq \frac{1 - \frac{(1+\delta)x_0}{A}}{x_0(1 + \delta)} - \frac{O(\eta(A - \rho)^2) \log(A/x_0)}{x_0^2 \log(1 + \delta)} \quad (98)$$

Proof. For every $g = 0, 1, \dots$, let T_g be the first iteration such that $x_t \geq (1 + \delta)^g x_0$. Let b be the smallest integer such that $(1 + \delta)^b x_0 \geq A$. Suppose for notation simplicity that we replace x_t with exactly A whenever $x_t \geq A$. By the definition of

T_g , we have

$$\sum_{t \in [T_g, T_{g+1})} \eta C_t [(1 + \delta)^g x_0]^2 \leq x_{T_{g+1}} - x_{T_g} \leq \delta(1 + \delta)^g x_0 + O(\eta(A - \rho)^2) \quad (99)$$

$$\sum_{t \in [T_g, T_{g+1})} \eta C_t [(1 + \delta)^{g+1} x_0]^2 \geq x_{T_{g+1}} - x_{T_g} \geq \delta(1 + \delta)^g x_0 - O(\eta(A - \rho)^2) \quad (100)$$

$$(101)$$

These imply that

$$\sum_{t \in [T_g, T_{g+1})} \eta C_t \leq \frac{\delta}{(1 + \delta)^g x_0} + \frac{O(\eta(A - \rho)^2)}{x_0^2} \quad (102)$$

$$\sum_{t \in [T_g, T_{g+1})} \eta C_t \geq \frac{\delta}{(1 + \delta)^{g+2} x_0} - \frac{O(\eta(A - \rho)^2)}{x_0^2} \quad (103)$$

Recall b is the smallest integer such that $(1 + \delta)^b x_0 \geq A$, so we can calculate

$$\sum_{t \geq 0, x_t \leq A} \eta C_t \leq \sum_{g=0}^{b-1} \frac{\delta}{(1 + \delta)^g x_0} + \frac{O(\eta(A - \rho)^2)}{x_0^2} b \quad (104)$$

$$= \frac{\delta}{1 - \frac{1}{1+\delta}} \frac{1}{x_0} + \frac{O(\eta(A - \rho)^2)}{x_0^2} b \quad (105)$$

$$= \frac{1 + \delta}{x_0} + \frac{O(\eta(A - \rho)^2)}{x_0^2} \frac{\log(A/x_0)}{\log(1 + \delta)} \quad (106)$$

$$\sum_{t \geq 0, x_t \leq A} \eta C_t \geq \sum_{g=0}^{b-2} \frac{\delta}{(1 + \delta)^{g+2} x_0} - \frac{O(\eta(A - \rho)^2)}{x_0^2} b \quad (107)$$

$$= \frac{\delta(1 + \delta)^{-1}(1 - \frac{1}{(1+\delta)^{(b-1)})})}{1 - \frac{1}{1+\delta}} \frac{1}{x_0} - \frac{O(\eta(A - \rho)^2)}{x_0^2} b \quad (108)$$

$$= \frac{1 - \frac{(1+\delta)x_0}{A}}{x_0(1 + \delta)} - \frac{O(\eta(A - \rho)^2)}{x_0^2} \frac{\log(A/x_0)}{\log(1 + \delta)} \quad (109)$$

Thus, the two desired inequalities are proved. $\square$

Lemma B.2 (Lemma D.19, (Allen-Zhu & Li, 2020)). *Let $\{x_t, y_t\}_{t=1, \dots}$ be two positive sequences that satisfy*

$$\begin{aligned} x_{t+1} &\geq x_t + \eta \cdot C_t (x_t - \rho)^2, \\ y_{t+1} &\leq y_t + S\eta \cdot C_t y_t^2, \end{aligned}$$

for some $C_t = \Theta(1)$. Suppose $x_0 \geq y_0 S \frac{1+2G}{1-3G}$ where $S \in (0, 1)$, $G \in (0, 1/3)$ and $0 < \eta \leq \min\{\frac{G^2 x_0}{\log(A/x_0)}, \frac{G^2 y_0}{\log(1/G)}\}$, $0 \leq \rho < O(x_0)$, and for all $A \in (x_0, O(1)]$, let T_x be the first iteration such that $x_t \geq A$. Then, we have $y_{T_x} \leq O(G^{-1} y_0)$.

Proof. Let T_x be the first iteration t in which $x_t \geq A$. Apply Lemma B.1 for the x_t sequence with $C_t = C_t$ and threshold A , we have

$$\sum_{t=0}^{T_x} \eta C_t \leq \frac{1 + \delta}{x_0} + \frac{O(\eta(A - \rho)^2)}{x_0^2} \frac{\log(A/x_0)}{\log(1 + \delta)} \quad (110)$$

$$= \frac{1 + \delta}{x_0} + O\left(\frac{\eta(A - \rho)^2 \log(A/x_0)}{\delta x_0^2}\right) \quad (111)$$

$$\leq \frac{1 + \delta}{x_0} + O\left(\frac{\eta \log(A/x_0)}{\delta x_0^2}\right) \quad (112)$$

Let T_y be the first iteration t in which $y_t \geq A$. Apply Lemma B.1 for the y_t sequence with $\eta = S\eta$, $C_t = C_t$, $\rho = 0$ and threshold $A' = G^{-1}y_0$, we have

$$\sum_{t=0}^{T_y} S\eta C_t \geq \frac{1 - \frac{(1+\delta)y_0}{A'}}{y_0(1+\delta)} - \frac{O(S\eta(A')^2) \log(A'/y_0)}{y_0^2 \log(1+\delta)} \quad (113)$$

$$\geq \frac{1 - O(\delta + G)}{y_0} - O\left(\frac{S\eta(A')^2 \log(1/G)}{\delta y_0^2}\right) \quad (114)$$

$$\geq \frac{1 - O(\delta + G)}{y_0} - O\left(\frac{S\eta \log(1/G)}{\delta y_0^2}\right) \quad (115)$$

Compare Equation 112 and 115. Choosing $\delta = G$ and $\eta \leq \min\{\frac{G^2 x_0}{\log(A/x_0)}, \frac{G^2 y_0}{\log(1/G)}\}$, together with $x_0 \geq y_0 S \frac{1+2G}{1-3G}$ we have $T_x \leq T_y$. $\square$

Lemma B.3 (Lemma K.15, (Jelassi & Li, 2022)). *Let $\{z_t\}_{t=0}^T$ be a positive sequence defined by the following recursions*

$$\begin{aligned} z_{t+1} &\geq z_t + m(z_t - \rho)^2, \\ z_{t+1} &\leq z_t + M(z_t)^2, \end{aligned}$$

where $z_0 > \rho \geq 0$ is the initialization and $m, M > 0$ are some constants. Let $v > z_0$, then the time T_v such that $z_{T_v} \geq v$ for all $t \geq T_v$ is

$$T_v = \frac{2z_0}{m(z_0 - \rho)^2} + \frac{4Mz_0^2}{m(z_0 - \rho)^2} \left\lceil \frac{\log(v/z_0)}{\log(2)} \right\rceil. \quad (116)$$

Proof. Let $n \in \mathbb{N}^*$. Let T_n be the first time that $z_t \geq 2^n z_0$. We want to find an upper bound of T_n . We start with the case $n = 1$. By summing the recursion, we have:

$$z_{T_1} \geq z_0 + m \sum_{t=0}^{T_1-1} (z_t - \rho)^2 \quad (117)$$

Because $z_t \geq z_0$, we obtain

$$T_1 \leq \frac{z_{T_1} - z_0}{m(z_0 - \rho)^2} \quad (118)$$

Now, we want to bound $z_{T_1} - z_0$. Using again the recursion and $z_{T_1-1} \leq 2z_0$, we have

$$z_{T_1} \leq z_{T_1-1} + M(z_{T_1-1})^2 \leq 2z_0 + 4Mz_0^2. \quad (119)$$

Combining Equation 118 and 119, we get a bound on T_1 as

$$T_1 \leq \frac{z_0 + 4Mz_0^2}{m(z_0 - \rho)^2} = \frac{z_0}{m(z_0 - \rho)^2} + \frac{4Mz_0^2}{m(z_0 - \rho)^2} \quad (120)$$

Now, let's find a bound for T_n . Starting from the recursion and using the fact that $z_t \geq 2^{n-1}z_0$ for $t \geq T_{n-1}$ we have

$$z_{T_n} \geq z_{T_{n-1}} + m \sum_{t=T_{n-1}}^{T_n-1} (z_t - \rho)^2 \quad (121)$$

$$\geq z_{T_{n-1}} + (2^{n-1})^2 m(z_0 - \rho)^2 (T_n - T_{n-1}) \quad (122)$$

On the other hand, by using $z_{T_{n-1}} \leq 2^n z_0$ we upper bound z_{T_n} as

$$z_{T_n} \leq z_{T_{n-1}} + M(z_{T_{n-1}})^2 \leq 2^n z_0 + M2^{2n} z_0^2 \quad (123)$$

Besides, we know that $z_{T_{n-1}} \geq 2^{n-1}z_0$. Therefore, we upper bound $z_{T_n} - z_{T_{n-1}}$ as

$$z_{T_n} - z_{T_{n-1}} \leq 2^{n-1}z_0 + M2^{2n} z_0^2 \quad (124)$$

Combining Equations 121 and 124 yields

$$T_n \leq T_{n-1} + \frac{2^{n-1}z_0 + M2^{2n}z_0^2}{(2^{n-1})^2m(z_0 - \rho)^2} \quad (125)$$

$$= T_{n-1} + \frac{z_0}{2^{n-1}m(z_0 - \rho)^2} + \frac{4Mz_0^2}{m(z_0 - \rho)^2} \quad (126)$$

Summing Equation 125 for $n = 2, \dots, n$ we have

$$T_n \leq \sum_{i=1}^n \frac{z_0}{2^{i-1}m(z_0 - \rho)^2} + \frac{4Mnz_0^2}{m(z_0 - \rho)^2} \leq \frac{2z_0}{m(z_0 - \rho)^2} + \frac{4Mnz_0^2}{m(z_0 - \rho)^2} \quad (127)$$

Lastly, we know that $2^n z_0 \geq v$ this implies that we can set $n = \left\lceil \frac{\log(v/z_0)}{\log(2)} \right\rceil$ in Equation 127. $\square$

We make the following assumptions for every $t \leq T$ as the same in (Jelassi & Li, 2022).

Lemma B.4 (Induction hypothesis D.1, (Jelassi & Li, 2022)). *Throughout the training process using GD for $t \leq T$, we maintain that, for every i and $j \in [J]$,*

$$|\langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle| \leq \tilde{O}(\sigma_0 \sigma_p \sqrt{d}). \quad (128)$$

Lemma B.5 (Lemma G.4, (Deng et al., 2023)). *For every i , we have $l_i^{(t)} = \Theta(1)g_1(t)$, where*

$$g_1(t) = \text{sigmoid} \left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3 \right). \quad (129)$$

Lemma B.6 (Lemma K.5, (Jelassi & Li, 2022)). *Let $X \in \mathbb{R}^d$ be a Gaussian random vector, $X \sim \mathcal{N}(0, \sigma^2 I_d)$. Then with probability at least $1 - o(1)$, we have $\|X\|_2^2 = \Theta(\sigma^2 \sqrt{d})$.*

Lemma B.7 (Lemma K.7, (Jelassi & Li, 2022)). *Let X and Y be independent Gaussian random vectors on $\mathbb{R}^d$ and $X \sim \mathcal{N}(0, \sigma^2 I_d)$, $Y \sim \mathcal{N}(0, \sigma_0^2 I_d)$. Assume that $\sigma \sigma_0 \leq \frac{1}{d}$. Then, with probability at least $1 - \delta$, we have*

$$|\langle X, Y \rangle| \leq \sigma \sigma_0 \sqrt{2d \log \frac{2}{\delta}}$$

Lemma B.8 (Bound on noise inner products). *Let $N = O(\text{poly}(d))$. The following hold with probability at least $1 - o(1)$:*

$$\begin{aligned} \max \left\{ |\langle \mathbf{w}_{j,\epsilon}^{(0)}, \boldsymbol{\xi}_i \rangle| \right\} &= \tilde{O}(\sigma \sigma_0 \sqrt{d}) \\ \max_i \left\{ \frac{1}{n} \sum_{k=1}^n |\langle \boldsymbol{\xi}_k, \boldsymbol{\xi}_i \rangle| \right\} &= \tilde{O}\left(\frac{\sigma^2 d}{N} + \sigma^2 \sqrt{d}\right) \end{aligned}$$

Proof. For the first inequality, Lemma B.7 implies that with probability at least $1 - \frac{1}{dN}$,

$$|\langle \mathbf{w}_{j,\epsilon}^{(0)}, \boldsymbol{\xi}_i \rangle| \leq \sigma \sigma_0 \sqrt{2d \log \left(\frac{2}{dN} \right)} = \tilde{O}(\sigma \sigma_0 \sqrt{d}) \quad (130)$$

Taking a union bound over $n = 1, \dots, N$ gives the result.

The second statement is proved similarly. $\square$

Lemma B.9 (Bound on the noise component for SAM). *Assume that $\rho = o(\sigma_0)$ and $\omega(1) \leq N \leq O(\text{poly}(d))$. Throughout the training process using SAM for $t \leq T$, we maintain that, for every i and $j \in [J]$,*

$$|\langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle| \leq \tilde{O}(\sigma \sigma_0 \sqrt{d}) \quad (131)$$

Proof. Let $\chi_t = \max\{|\langle \mathbf{w}_{j,\epsilon}^{(t)}, \boldsymbol{\xi}_i \rangle|\}$, $\alpha = \max_i \{\frac{1}{n} \sum_{k=1}^n |\langle \boldsymbol{\xi}_k, \boldsymbol{\xi}_i \rangle|\}$. Combined with $l_{k,\epsilon}^{(t)} \leq 1$, the noise gradient update rule can be bounded as

$$\chi_{t+1} \leq \chi_t + 3\eta\alpha\chi_t^2$$

Suppose that $a(t)$ satisfies the differential equation

$$\begin{aligned} a' &= 3\eta\alpha a^2 \\ a(0) &= \chi_0 \end{aligned}$$

Observe that $a(t)$ is increasing so, by the Mean Value Theorem there exists $\tau \in (t, t+1)$ such that

$$\begin{aligned} a(t+1) - a(t) &= a'(\tau) \\ &= 3\eta\alpha a(\tau)^2 \\ &\geq 3\eta\alpha a(t)^2 \end{aligned}$$

So an easy induction shows that $a(t) \geq \chi_t$.

Now solving for $a(t)$,

$$a(t) = \frac{1}{\frac{1}{a(0)} - 3\eta\alpha t}, \quad t \leq \frac{1}{3\eta\alpha a(0)}.$$

Using the high probability tail bounds B.8,

$$\begin{aligned} a(0) &= \chi_0 = \tilde{O}(\sigma\sigma_0\sqrt{d}) \\ \alpha &= \tilde{O}\left(\frac{\sigma^2 d}{N} + \sigma^2\sqrt{d}\right) \end{aligned}$$

where $\sigma = \frac{\sigma_p}{\sqrt{d}}$. Substituting these bounds gives

$$a(t) = \frac{1}{\tilde{\Omega}\left(\frac{1}{\sigma\sigma_0\sqrt{d}}\right) - \tilde{O}\left(\eta t\left(\frac{\sigma^2 d}{N} + \sigma^2\sqrt{d}\right)\right)}$$

Now Theorem A.15 and $\rho = o(\sigma_0)$ implies that $\eta T_0 = \tilde{\Theta}\left(\frac{1}{\sigma_0\beta_e^3}\right)$. Combined with the assumption that $N = \Omega(1)$, the second term in the denominator is of lower order than the first term, so

$$a(T_0) = \tilde{O}(\sigma\sigma_0\sqrt{d}).$$

We conclude that

$$|\langle \mathbf{w}_j^{(t)}, \boldsymbol{\xi}_i \rangle| \leq \tilde{O}(\sigma_0\sigma_p\sqrt{d}). \quad (132)$$

□

Lemma B.10. For every i , we have $l_i^{(t)} = \Theta(1)g_1(t)$ and $l_{i,\epsilon}^{(t)} = \Theta(1)g_2(t)$, where

$$g_1(t) = \text{sigmoid}\left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_j^{(t)}, \mathbf{v}_e \rangle^3\right), \quad (133)$$

$$g_2(t) = \text{sigmoid}\left(\sum_{j=1}^J -\beta_d^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_d \rangle^3 - \beta_e^3 \langle \mathbf{w}_{j,\epsilon}^{(t)}, \mathbf{v}_e \rangle^3\right). \quad (134)$$

The proof is the same as (Deng et al., 2023)[Lemma G.4].

C. Additional Experimental Settings

C.1. Datasets and Training Details

Datasets. The CIFAR10 dataset (Krizhevsky et al., 2009) consists of 60,000 32×32 color images in 10 classes, with 6000 images per class. The CIFAR100 dataset (Krizhevsky et al., 2009) is just like the CIFAR10, except it has 100 classes containing 600 images each. For both of these datasets, the training set has 50,000 images (5,000 per class for CIFAR10 and 500 per class for CIFAR100) with the test set having 10,000 images. CINIC10 (Darlow et al., 2018) represents an image classification dataset consisting of 270,000 images, which is 4.5 times larger than CIFAR10. The dataset is created by merging CIFAR10 with images extracted from the ImageNet database, specifically selecting and downsampling images from the same 10 classes present in CIFAR10. Tiny-ImageNet (Le & Yang, 2015) comprises 100,000 images distributed across 200 classes of ImageNet (Deng et al., 2009), with each class containing 500 images. These images have been resized to 64×64 dimensions and are in color. The dataset consists of 500 training images, 50 validation images, and 50 test images per class. The STL10 dataset (Coates et al., 2011) includes 5000 96×96 training labeled images, 500 per CIFAR10 class. The test set consists of 800 images per class, this counts up to 8,000 images in total.

Training on different datasets. Follow the setting from (Andriushchenko & Flammarion, 2022; Andriushchenko et al., 2023), we trained Pre-Activation ResNet18 on all datasets except for CIFAR100 which was trained with ResNet34. We trained our models for 200 epochs with a batch size of 128 and used basic data augmentations such as random mirroring and random crop. We used SGD with the momentum parameter of 0.9 and set weight decay to 0.0005. We also fixed $\rho = 0.1$ for SAM unless further specified. For all datasets, we used a learning rate schedule where we set the initial learning rate to 0.1. The learning rate is decayed by a factor of 10 after 50% and 75% epochs, i.e., we set the learning rate to 0.01 after 100 epochs and to 0.001 after 150 epochs.

Training with different architectures. We used the same training procedures for Pre-Activation ResNet18, VGG19, and DenseNet121. We directly used the official Pytorch (Paszke et al., 2019) implementation for VGG19 and DenseNet121. For 3-layer MLPs, we used a hidden size of 512 with a dropout of 0.1 to avoid overfitting and set $\rho = 0.01$. For ViT-S (Yuan et al., 2021), we adopted a Pytorch implementation at <https://github.com/lucidrains/vit-pytorch>. In particular, the hidden size, the depth, the number of attention heads, and the MLP size are set to 768, 8, 8, and 2304, respectively. We adjusted the patch size to 4 to fit the resolution of CIFAR10 and set both the initial learning rate and ρ to 0.01.

Computational resources. Each model is trained on 1 NVIDIA RTX A5000 GPU.

C.2. Other Implementation Details

When to separate the examples? We selected the best-separating epoch t in the set of $\{4, 5, 6, 7, 8, 10\}$ for CIFAR10 and $\{12, 14, 16, 18, 20, 22\}$ for CIFAR100. Particularly, we separated examples of CIFAR10 at around epoch 8 while that of CIFAR100 is near epoch 20. Near this point, the gain in training error diminishes significantly as shown in Figure 11a, which shows a sign that the model successfully learns easy features. In addition, we reported the results for different separating epochs in Appendix D.6.

Forgetting score. To compute forgetting scores of training examples in each dataset, we collected the same statistics as in (Toneva et al., 2018) but computed at the end of each epoch. The reason is to make the statistics consistent between two versions of the same difficult example which is repeated in the upsampled dataset.

Hessian spectra. We approximated the density of the Hessian spectrum using the Lanczos algorithm (Papayan, 2018; Ghorbani et al., 2019). The Hessian matrix is approximated by 1000 examples (100 per class of CIFAR10). Then we extract the top eigenvalues to calculate the maximum Hessian eigenvalue ($\lambda_{\max}$) and the bulk of spectra ($\lambda_{\max}/\lambda_5$) (Jastrzebski et al., 2020).

D. Additional Results

D.1. An Extreme Case of Toy Datasets

We consider an extreme setting when two features have identical strength and no missing easy features, i.e., $\beta_e = \beta_d = \alpha = 1$, the gap between LHS and RHS in Equation 11 is not small as shown in the figure 6. The gap is consistently around 0.2-0.3 from epoch 250 onwards.

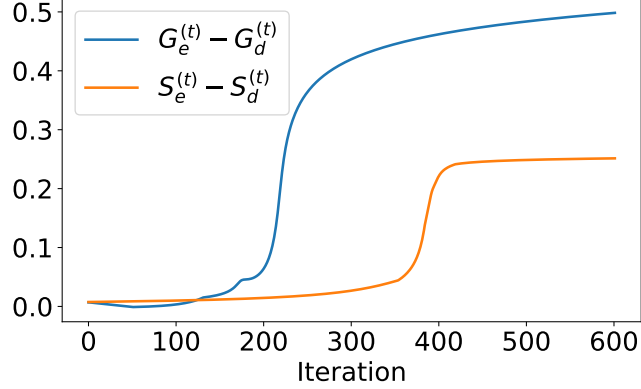


Figure 6: The gap between contribution of easy and difficult features towards the model output in SAM and GD. The toy datasets is generated from the distribution in Definition 3.1 with $\beta_d = \beta_e = \alpha = 1$.

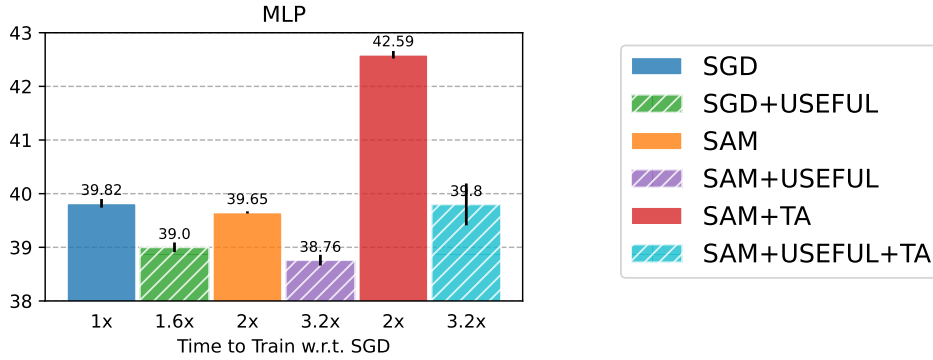


Figure 7: **Test classification errors of 3-layer MLP on CIFAR10.** The number below each bar indicates the approximated cost to train the model and the tick on top shows the standard deviation over three runs. USEFUL improves the performance of SGD and SAM when training with 3-layer MLP.

Table 4: Test classification errors for training SAM and ASAM on the original CIFAR10 and modified datasets by USEFUL. Results are averaged over 3 seeds.

	SAM	ASAM
+	4.23 ± 0.08	4.33 ± 0.19
+ USEFUL	4.04 ± 0.06	4.09 ± 0.10
+ TA	4.06 ± 0.08	3.93 ± 0.11
+ TA + USEFUL	3.49 ± 0.09	3.46 ± 0.01

D.2. USEFUL is Useful for MLP

Figure 7 shows that for 3-layer MLP, USEFUL successfully reduces the test error of both SGD and SAM by nearly 1%. Additionally, SGD+USEFUL yields better performance than SAM alone.

D.3. SAM & USEFUL Reduce Forgetting Scores

We used forgetting scores (Toneva et al., 2018) to partition examples in CIFAR10 into different groups. Forgetting scores count the number of times an example is misclassified after being correctly classified during training and is an indicator of the learning-difficulty of examples. Figure 8 illustrates that SGD+USEFUL and SAM have fewer examples with high forgetting scores than SGD does. This aligns with our theoretical analysis in Theorem 3.4 and results on the toy datasets. By upsampling difficult examples in the dataset, they contribute more to learning and hence SGD+USEFUL learns difficult features faster than SGD.

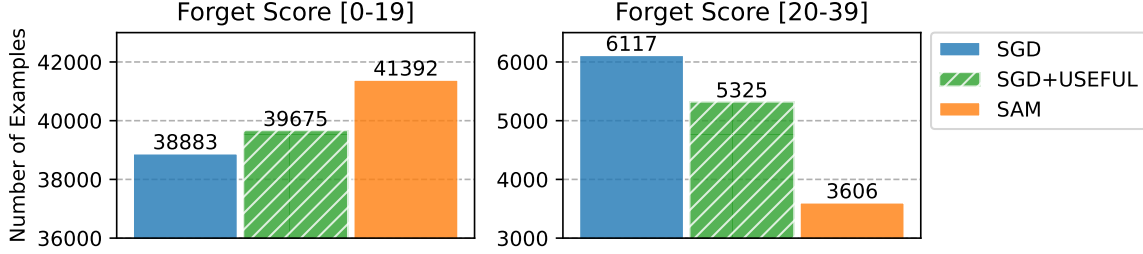


Figure 8: **Forgetting scores for training ResNet18 on CIFAR10.** Forgetting scores measure the learning difficulty of examples in training data. USEFUL approaches the training dynamics of SAM, with more examples being forgotten infrequently and fewer examples being forgotten frequently.

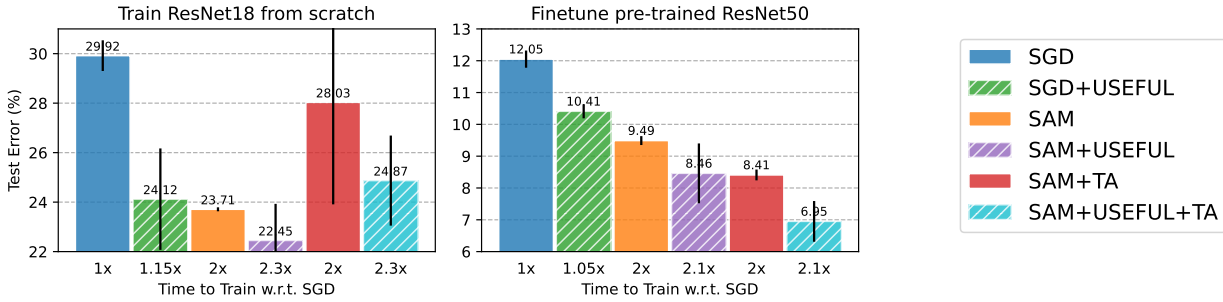


Figure 9: **Comparing test classification errors on Waterbirds.** The number below each bar indicates the approximated cost to train the model and the tick on top shows the standard deviation over three runs. USEFUL boosts the performance of SGD and SAM on the balanced test set, showing its generalization to the OOD setting. In addition, the success of USEFUL in fine-tuning reveals its new application to the transfer learning setting.

Table 5: Comparison between USEFUL and upweighting loss regarding test classification errors. Upweighting loss doubled the loss for all examples in the difficult clusters found by USEFUL, which is different from dynamically upweighting examples during the training (Zhai et al., 2022). Results are averaged over 3 seeds.

METHOD	SGD		SAM	
	CIFAR10	CIFAR100	CIFAR10	CIFAR100
UPWEIGHTING LOSS	5.33 ± 0.09	26.70 ± 3.25	4.28 ± 0.02	21.75 ± 0.25
USEFUL	4.79 ± 0.05	22.58 ± 0.08	4.04 ± 0.06	20.66 ± 0.44

D.4. USEFUL generalizes to other SAM variants.

In this experiment, we show that SAM can also generalize to other variants of SAM. We chose ASAM, which is proposed by Kwon et al. to address the sensitivity of parameter re-scaling (Dinh et al., 2017). Following the recommended settings in ASAM, we trained it with a perturbation radius $\rho = 1.0$, which is 10 times that of SAM. Other settings are identical to the standard settings in Appendix C. Table 4 demonstrates the results for training ResNet18 on CIFAR10. Both SAM and ASAM can be combined with USEFUL to improve the test classification error. When using TA, ASAM shows a slightly better performance than SAM.

D.5. USEFUL generalizes to OOD

While our main contribution is providing a novel and effective method to improve the in-distribution generalization performance, we conducted new experiments confirming the benefits of our method to improving out-of-distribution (OOD) generalization performance. As a few very recent works (Cha et al., 2021; Wang et al., 2023) showed the benefit of SAM in improving OOD performance, it is expected that USEFUL also extend to this setting. As can be seen in Figure 9, both

Table 6: **Test classification errors of SGD** for different partition methods. Results are averaged over 3 seeds.

PARTITION METHOD	CIFAR10	CIFAR100
QUANTILE	5.27 ± 0.10	23.49 ± 0.82
MISCLASSIFICATION	4.98 ± 0.17	23.86 ± 0.70
USEFUL	4.79 ± 0.05	22.58 ± 0.08

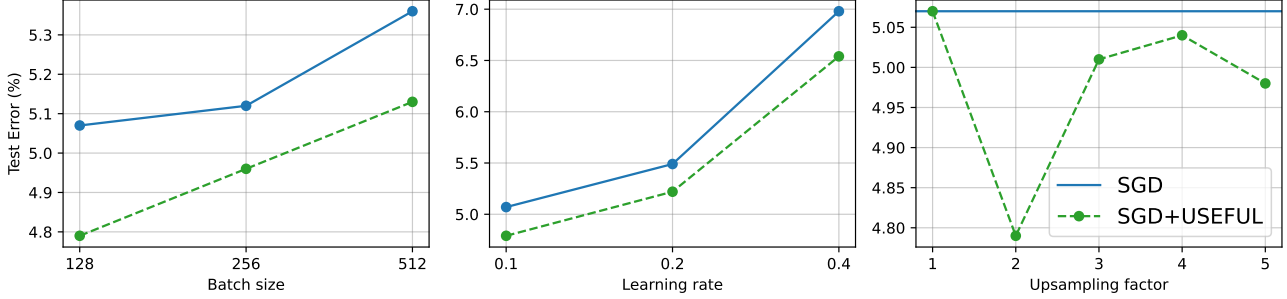


Figure 10: **Ablation studies of training ResNet18 on CIFAR10.** In each experiment, we used the standard training settings while (left) varying training batch size or (middle) varying learning rate, or (right) varying upsampling factor.

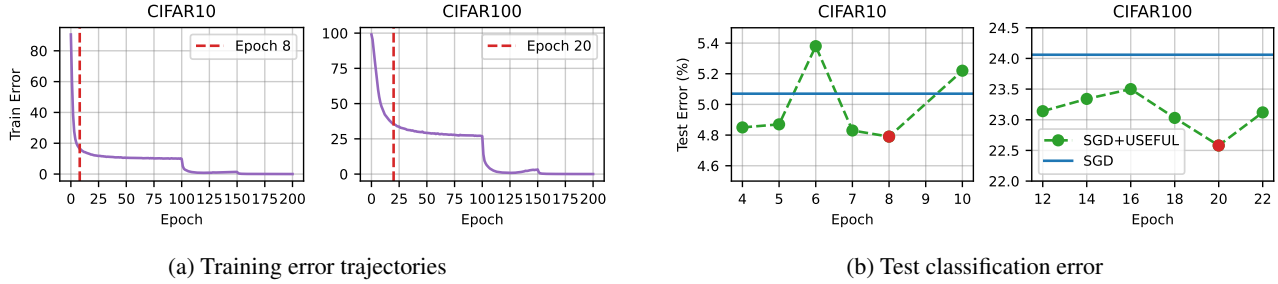


Figure 11: **Separating epoch analysis.** (left) Red lines indicate our optimal choice of t to separate examples at and restart training. The early epoch that can best separate the examples is when the change in training error starts to shrink. (right) Red points indicate our optimal choice of t .

SAM and USEFUL effectively improve the performance on the balanced test set though the model is trained on the spurious training set. When training ResNet18 with SGD from scratch, USEFUL decreases the classification errors by 5.8%. In addition, it can be successfully applied to the pre-trained ResNet50 (on ImageNet), opening up a promising application to the transfer learning setting.

D.6. Ablation studies

USEFUL vs Upweighting loss. We compare USEFUL with upweighting loss of examples in the difficult clusters. As can be seen in Table 5, when coupling with either SGD or SAM, USEFUL clearly outperforms upweighting loss on both CIFAR datasets. It is worth mentioning that upweighting loss is different from iteratively importance sampling methods such as GRW (Zhai et al., 2022), which dynamically upweights examples during the training by factors that depend on the loss value. In addition, GRW is dedicated to the distribution shift setting while our paper considers the in-distribution setting.

Data selection method. In this experiment, we compare clustering with other methods for partitioning data. The first baseline is to upsample misclassified examples (MISCLASSIFICATION) while the second baseline is to upsample all examples whose training errors are larger than the median value (QUANTILE). All the three methods are performed at the same epoch t . Table 6 shows that USEFUL selects a better set of upsampling examples, leading to the best model performance.

Training batch size. Figure 10 left shows the gap between USEFUL and SGD when changing the batch size. Our method consistently improves the performance, proving its effectiveness is not simply captured by the gradient variance caused by

the training batch size.

Learning rate. The small learning rate is a nuance in our theoretical results to guarantee that easy and difficult features are separable in early training. In general, a small learning rate is required for most theoretical results on gradient descent and its convergence and is a standard theoretical assumption (Roux et al., 2012; Schmidt et al., 2017; Wen et al., 2022; Chen et al., 2023). In practice, both for separating easy vs difficult examples and for training on the upsampled data, we used the standard learning rate that results in the best generalization performance for both SGD and SAM following prior work (Kwon et al., 2021; Zheng et al., 2021; Andriushchenko & Flammarion, 2022). While the theoretical requirement for the learning rate is always smaller than the one that is used in practice, empirically a larger learning rate does not yield better generalization for the problems we considered in contrast to other settings (Li et al., 2019; Puli et al., 2024). As shown in Figure 10 middle, increasing the learning rate has an adverse effect on the model performance. Indeed, for a fair comparison, the algorithms should be trained with hyperparameters that empirically yield the best performance; otherwise, the conclusions are not valid. Nevertheless, USEFUL always improves the test error across different learning rates.

Upsampling factor. We empirically found the upsampling factor of 2 to consistently work well across different datasets and architectures. Using larger upsampling factors results in a too-large discrepancy between the training and test distribution and does not work better, as is expected and discussed in Section 1. As illustrated in Figure 10 right, while all factors from 2 to 5 bring performance improvement, the upsampling factor of 2 yields the best performance, as it reduces the simplicity bias with minimum change to the rest of the data distribution.

Separating epoch. Fig 11a shows that the early epoch that can best separate the examples is when the change in training error starts to shrink. At this time, examples that are learned can be separated from the rest by clustering model outputs as analyzed in our theoretical results in Section 3.3. Figure 11b demonstrates the performance of USEFUL when separating examples at different epochs in early training. Too early or too late epochs do not cluster examples well, i.e., some examples with easy features fall into the difficult clusters and vice versa. This ablation study shows that upsampling correct examples and with enough amount is important for our method to achieve its best.