HOW TO FINE-TUNE VISION MODELS WITH SGD

Anonymous authors
Paper under double-blind review

ABSTRACT

SGD (with momentum) and AdamW are the two most commonly used optimizers for fine-tuning large neural networks in computer vision. When the two methods perform the same, SGD is preferable because it uses less memory and is more efficient than AdamW. However, when evaluating on downstream tasks that differ significantly from pretraining, we find that across five popular benchmarks SGD fine-tuning gets substantially lower accuracies than AdamW on many modern vision models such as Vision Transformers and ConvNeXts—especially out-of-distribution (OOD). We find that such large gaps arise in instances where the fine-tuning gradients in the first ("embedding") layer are much larger than the rest of the model. Our analysis suggests an easy fix: if we simply freeze the embedding layer (0.7% of the parameters), SGD performs competitively with AdamW while using less memory across a suite of benchmarks. Our insights lead to state-of-the-art accuracies on popular distribution shift benchmarks: WILDS-FMoW, WILDS-Camelyon, BREEDS-Living-17, Waterbirds, and DomainNet.

1 INTRODUCTION

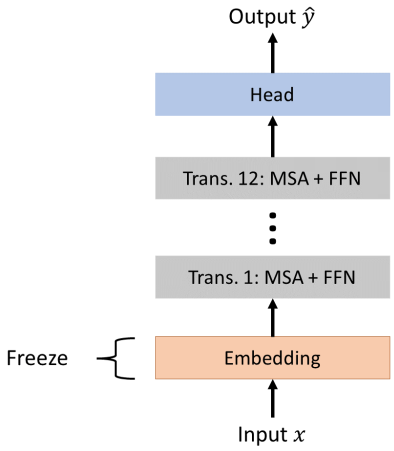
Finetuning large pretrained models on downstream tasks has become a dominant approach in deep learning (Kornblith et al., 2019; Chen et al., 2020; Zhai et al., 2020). The two most commonly used optimizers in current practice of deep learning are SGD and AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019)[1]. While most modern neural network architectures increasingly use AdamW for pretraining, it is still common to use SGD for fine-tuning, especially in computer vision (Dosovitskiy et al., 2021; Steiner et al., 2021; Kumar et al., 2022). Part of the appeal is that SGD is more memory and compute efficient (Adam needs 2x model weights) (Ginsburg et al., 2019). In large models requiring multi-gpu inference/training, the additional memory requirement can double the number of GPUs required. Additionally, papers that do compare AdamW and SGD find that they perform comparably on ImageNet like domains (Touvron et al., 2021). In contrast, we reach different conclusions when finetuning modern vision models on datasets that differ significantly from pretraining data and/or have substantial distribution shifts.

We examine seven popular models—including Vision Transformers (Dosovitskiy et al., 2021), ConvNeXt (Liu et al., 2022), ResNets (He et al., 2016) of different sizes—on 5 standard distribution shift datasets (WILDS-FMoW, WILDS-Camelyon, Waterbirds, BREEDS-Living-17, DomainNet). These datasets were selected to capture different types of data shifts (subpopulation shifts, spurious correlations, style shifts), including two real world shifts in medical imaging and satellite remote sensing from the WILDS benchmark (Koh et al., 2021).

We find that for older models (ResNet-50 and ResNet-101) there is a minimal difference between SGD and AdamW, while for newer models like vision transformers and ConvNeXt, AdamW can have significantly higher accuracy, especially out-of-distribution (OOD). For example, on a CLIP base Vision Transformer (ViT-B/16), fine-tuning with AdamW gets 3.3% higher accuracy ID and 8.8% higher accuracy OOD (Figure 1b). These gains are consistent—averaged across all models and datasets, fine-tuning with AdamW gets 1.3% higher accuracy ID (Table 2) and 3.9% higher accuracy OOD (Table 1).

A key difference between AdamW and SGD, is that AdamW divides the gradient update for each parameter by the running second moment of the gradient—parameters with consistently high gra-

[1]We use SGD and AdamW to narrowly refer to stochastic gradient descent with momentum and Adam with decoupled weight decay, respectively. This differs from the terminology in the broader optimization literature.



(a) Simplified schematic of a Vision Transformer, where we freeze the first ("embedding") layer.

	ID Acc.	OOD Acc.
SGD	89.8%	67.2%
AdamW	(+3.3%)	(+8.8%)
SGD (Freeze-Embed)	(+3.3%)	(+8.7%)
Gradual-Unfreeze	**(+3.5%)**	**(+9.0%)**

(b) Performance of different fine-tuning methods on a CLIP-ViT-B/16 averaged over 5 distribution shifts.

Figure 1: We fine-tune 7 models (including Vision Transformers, ConvNeXt, ResNet) on 5 distribution shift datasets (BREEDS-Living-17, Waterbirds, DomainNet, WILDS-Camelyon, WILDS-FMoW). Fine-tuning with SGD + momentum gets lower accuracies than AdamW on modern architectures (Vision Transformers and ConvNeXt), especially out-of-distribution (OOD). Interestingly, a minor tweak to SGD where we "freeze" the first layer ($\approx 0.7\%$ of parameters—see e.g., Figure 1a) is competitive with AdamW. Gradually unfreezing a model's layers using a carefully annealed learning rate can sometimes further improves accuracies, while reducing computation time by around 30%. We give results for all datasets and models in our paper.

dients will change less when using AdamW than with SGD. Towards understanding why AdamW outperforms SGD, we examine the gradients at each layer of the deep learning models (Figure 1a). We find that for the models where AdamW outperforms SGD (Vision Transformers and ConvNeXt), the gradients of the first ("embedding") layer are much larger than the gradients of the other layers. For the other models (ResNet-50 and ResNet-101), the gradient of the embedding layer is *smaller* than the other layers.

To test if the embedding layer is in fact why SGD performs worse than AdamW, we consider a minor modification where we freeze the embedding layer and tune the rest of the model with SGD (Figure 1a)—we call this SGD (Freeze-embed). In models like base Vision Transformers, the embedding layers is around 0.7% of the total parameters of the model, so a priori we might not expect a substantial difference in accuracies. Interestingly, freezing the embedding layer leads to competitive accuracies with AdamW (Figure 1b). Averaged across all datasets and models, SGD-Freeze embed gets 76.7% accuracy OOD (vs. 71.9% for SGD and 76.0% for AdamW).

Inspired by our results from freezing embedding layer, we revisit the layerwise unfreezing technique proposed in prior work like ULMFiT (Howard & Ruder, 2018) and related methods (Mukherjee & Awadallah, 2019; Romero et al., 2020). We find that with a certain exponential decay in learning rate, a gradual unfreezing procedure in some cases can lead to further gains ID and OOD (Figure 1b). However, a large fraction of the gain over SGD comes from simply freezing the embedding layer, which suggests that the embedding layer plays a key role in modern vision architectures such as Vision Transformers and ConvNeXt.

These methods and insights, while simple, lead to state-of-the-art accuracies on all five datasets: WILDS-Camelyon, WILDS-FMoW, DomainNet, Waterbirds, and BREEDS Living-17, while being more efficient than AdamW.

2 Setup and Motivation

We use the following notation for deep networks. A network map $f_\theta : \mathcal{X} \to \mathcal{Y}$ is represented as a the composition of layers as $f_\theta = f^{(\mathsf{head})}_{\theta_{\mathsf{head}}} \circ f^{(\mathsf{L})}_{\theta_L} \circ \ldots \circ f^{(1)}_{\theta_1} \circ f^{(\mathsf{embed})}_{\theta_{\mathsf{embed}}}$, where

$\theta = (\theta_{\mathsf{head}}, \theta_L, \ldots, \theta_1, \theta_{\mathsf{embed}})$ denote all the parameters of the model. We use $f^{(\mathsf{embed})}_{\theta_{\mathsf{embed}}}$ and $f^{(\mathsf{head})}_{\theta_{\mathsf{head}}}$ to denote blocks of the network that are conceptually considered the "embedding layer" and the "head", respectively; and $\{f^{(1)}_{\theta_1}, \ldots, f^{(\mathsf{L})}_{\theta_L}\}$ denote the intermediate layers.

Fine-tuning. Consider networks that have been *pretrained* to get an initialization $\theta^{\mathsf{pretrain}}$. We focus on *fine-tuning* such models on a labeled dataset $D_{\mathsf{train}} \sim P_{\mathsf{finetune}}$ from a new task. Concretely, given a loss function $l : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$, we minimize the training loss using iterative optimization algorithms starting from the initialization $\theta^{\mathsf{pretrain}}$:

$$\min_{\theta} L_{\mathsf{train}}(\theta) = \frac{1}{|D_{\mathsf{train}}|} \sum_{(x,y) \in D_{\mathsf{train}}} l(f_\theta(x), y), \text{ with initialization } \theta \leftarrow \theta^{\mathsf{pretrain}}. \tag{2.1}$$

Optimizers. To minimize $L_{\mathsf{train}}(\theta)$ from initialization $\theta^{\mathsf{pretrain}}$, the two most common optimizers in deep learning are SGD (with momentum) and AdamW. AdamW is an adaptive gradient method that normalizes the gradient update for each parameter using a rolling average of the second moment statistics of the gradient, which doubles the optimizer memory. We use the standard implementations in PyTorch.

Evaluation. We evaluate accuracy of finetuned models on held-out in-distribution (ID) examples $D_{\mathsf{id}} \sim P_{\mathsf{finetune}}$, sampled from the same distribution as fine-tuning. To examine whether we have learned a robust model, we also measure accuracy on out-of-distribution (OOD) examples $D_{\mathsf{ood}} \sim P_{\mathsf{ood}}$ on standard benchmarks.

2.1 Datasets and model architectures

Datasets. We choose five fine-tuning benchmarks that capture different types of data shifts (sub-population shifts, spurious correlations, style shifts), including two real world shifts in medical imaging and satellite remote sensing from the WILDS benchmark. To examine whether we have learned a robust model, in addition to the in-distribution (ID) accuracy, we also measure accuracy on structured out-of-distribution (OOD) test sets for the benchmarks. The datasets we consider are:

1. **Living-17** (Santurkar et al., 2020) is a sub-population shift dataset from the BREEDS benchmark (Santurkar et al., 2020). The goal is to classify an image as one of 17 animal categories such as "bear"—for example, the ID dataset contains images of black bears and sloth bears and the OOD dataset has images of brown bears and polar bears.
2. **Waterbirds** (Sagawa et al., 2020) is a spurious correlation dataset where the goal is to classify an image as a "waterbird" or "landbird". In the ID dataset, "water" backgrounds are typically correlated with "waterbird" labels in the ID, but anticorrelated OOD.
3. **DomainNet** (Peng et al., 2019) is a standard domain adaptation dataset. Here, our ID dataset contains "sketch" images (e.g., drawings of apples, elephants, etc), and the OOD dataset contains "real" images of the same categories. We use the version of the dataset from Tan et al. (2020).
4. **WILDS-FMoW** (Christie et al., 2018; Koh et al., 2021): the goal is to classify a satellite image into one of 62 categories such as "impoverished settlement" or "hospital". The ID dataset contains satellite images from across the world between 2002 and 2012, and the OOD dataset contains images from Africa in 2017.
5. **WILDS-Camelyon** (Bandi et al., 2018; Koh et al., 2021): the goal is to classify whether a tissue slide has a tumor or not. The ID and OOD datasets contain slides from different hospitals.

Model Architectures. We consider seven popular models that span different architectures (vision transforms, convolutional networks), network sizes, and pretraining objectives (multi-modal, supervised, self-supervised): CLIP-base vision transformer (CLIP ViT-B/16) (Radford et al., 2021), CLIP-large vision transformer (CLIP ViT-/14), Imagenet pretrained base vision transformer (ViT-B/16) (Dosovitskiy et al., 2021), DINO base vision transformer (DINO ViT-B/16) (Caron et al., 2021), ConvNeXt base (ConvNeXt-B) (Liu et al., 2022), BIT pretrained ResNet-50 (BiT-RN50), BIT pretrained ResNet-101 (BiT ResNet101) (Kolesnikov et al., 2020).

Training protocols. For all our experiments, we use a cosine learning rate schedule and sweep over 6 initial learning rates. For all datasets we follow prior work (Kumar et al., 2022) and pick the best learning rate and early stop based on the ID validation accuracy, since we want to measure robustness to unseen OOD examples.

3 PRELIMINARY EXPERIMENTS: SGD, ADAMW, AND LAYER GRADIENTS

3.1 SGD VS. ADAMW

AdamW outperforms SGD. We find that fine-tuning with AdamW gets higher accuracy than fine-tuning with SGD (even with grid-searched learning rate), especially out-of-distribution (OOD)—see table 2 for the full ID results and table 1 for the full OOD results. Averaged across the 7 models and 5 datasets, AdamW gets an average in-distribution (ID) accuracy of 91.5% (vs. 90.2% for SGD), and an average OOD accuracy of 76.0% (vs. 72.1% for SGD). We note that this happens even though we sweep over 6 learning rates, use cosine learning rate decay, and early stop for SGD.

Accuracy gap increases with depth. The gains of AdamW increase with depth: AdamW gets on average 8.7% higher OOD accuracy on a CLIP base vision transformer (ViT-B/16), but 13.6% higher OOD accuracy on a CLIP large vision transformer (ViT-L/14). This suggests that the difference between these optimizers can play an even larger role as we scale up to bigger models.

AdamW $\approx$ SGD on ResNets. Breaking down the results by model type, we find that AdamW and SGD perform comparably for older ResNet architectures (ResNet-50 and ResNet-101). For example, on a ResNet-101, AdamW gets an average OOD accuracy of 74.5% (vs. 74.6% for SGD). However, for modern architectures such as Vision Transformers and ConvNeXt, AdamW gets higher accuracies.

3.2 EXAMINING LAYER GRADIENTS

AdamW normalizes gradients. Recall that AdamW divides the gradient update for each parameter by the running second moment of the gradient. So parameters with consistently high gradients will change less when using AdamW than with SGD. This suggests looking at the magnitude of the gradients across different components of the neural network—if they vary a lot, then AdamW and SGD will behave very differently. Towards understanding why AdamW outperforms SGD, we examine the gradients at each layer of the deep learning models.

Measure gradient norm at each layer. We measure the average norm of the gradient at each layer, across minibatches of the training set. More formally, at each step, SGD samples a random minibatch $B = \{(x_1, y_1), \ldots, (x_b, y_b)\}$ from the training set, and updates the parameters based on the gradient on the minibatch. The gradient at layer i for minibatch B is given by the average of the gradient for each example x_i, y_i in the minibatch:

$$g_i(B) = \frac{1}{|B|} \sum_{i=1}^{|B|} \nabla_{\theta_i} l(f_{\theta^{\mathsf{pretrain}}}(x_i), y_i). \tag{3.1}$$

The norm of the gradient, $\|g_i(B)\|_2$ roughly measures how much layer i will change after one step of SGD (its not exact as SGD also uses momentum). We compute the average of the gradient norms across all minibatches $B_1, \ldots, B_m$ in the training set D_{train} at the beginning of finetuning:

$$g_i = \frac{1}{m} \sum_{i=1}^{m} \|g_i(B_i)\|_2. \tag{3.2}$$

We measure this for each layer, including the embedding and head layers, at the pretrained initialization $\theta^{\mathsf{pretrain}}$—without actually performing any gradient updates.

First-layer gradient is an outlier. We show the gradient norm on DomainNet for each model in Figure 2, and include plots for the other datasets in Appendix A. We note the the gradient of the last ("head") layer, shown in green, is usually high because it is randomly initialized (Kumar et al.,

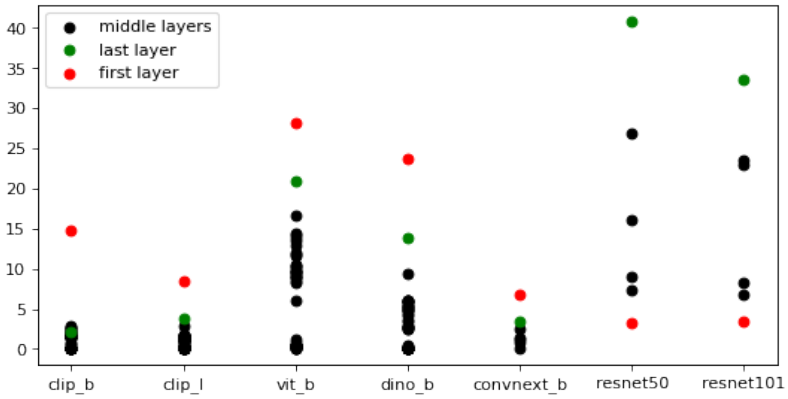


Figure 2: We visualize the norm of the gradients at different layers of each neural network, on DomainNet. The first ("embedding") layer is shown in red and an outlier in all cases, but is *smaller* than other layers for ResNet models, and *larger* for the Vision Transformers and ConvNeXt. The middle layers are shown in black, and the head in green. For transformer models, we have separate (black) points for the MLP and the multi-head self-attention layers.

2022) while the rest of the model is pretrained. The first ("embedding") layer is shown in red, and is an outlier in all cases—but it behaves differently for different models. For Vision Transformers and ConvNeXt, the gradients of the first layer (red) are much *higher* than the other layers—for these architectures, AdamW outperforms SGD. For the two ResNet models, the gradient of the embedding layer is *smaller* than the other layers. This reflects the common wisdom that lower layers learn more transferable features and do not need to change much when fine-tuning—for these architectures AdamW and SGD perform comparable.

Freezing layers. While Figure 2 correlates with the performance gaps between AdamW and SGD, it is not clear that the embedding-layer gradient *causes* SGD to perform poorly. To test this out, in the next Section we consider freezing the embedding layer and fine-tuning the rest of the model with SGD. The embedding layer only consists of a small fraction of the model parameters (e.g., 0.7% in a base-vision transformer), so a priori freezing the embedding layer is a very tiny tweak—and we would not expect a large change in accuracy.

4 Detailed experiments on freeze-embedding

SGD (Freeze-embed). In Section 2, we saw that the norm of the gradient of the first ("embedding") layer was much larger when AdamW outperformed SGD. To test if the embedding layer is really responsible for the poor performance of SGD, we consider freezing the embedding layer and only perform gradient updates on the rest of the network—we call this method "SGD (Freeze-embed)".

Experiment protocols. We fine-tune with each method—SGD, AdamW, and SGD (Freeze-embed)—for the same number of epochs, using a cosine learning rate schedule, and sweep over 6 learning rates. We check that the optimal learning rate is in the middle of the sweep to ensure that the performance gaps are not because we chose too small or too large a learning rate. For more details, see Appendix A.

Overall results. Table 2 shows the in-distribution (ID) accuracy on held out examples sampled from the same distribution as the fine-tuning data, and Table 1 shows the out-of-distribution (OOD) accuracy. Averaged across all the models and datasets, SGD (Freeze-embed) performs the best OOD, getting an average OOD accuracy of 76.7% (vs. 71.9% for SGD and 76.0% for AdamW). On ID data, SGD (Freeze-embed) closes 85% of the gap between SGD and AdamW, getting an average accuracy of 91.3% (vs. 90.2% for SGD and 91.5% for AdamW). We emphasize that we aren't claiming that SGD (Freeze-embed) is a substantially better method than AdamW. What we

Table 1: **Out-of-distribution (OOD)** accuracies for fine-tuning 7 popular models across 5 benchmark datasets, where we fine-tune with SGD, AdamW, or SGD (Freeze-Embed). AdamW gets higher accuracies than SGD for modern architectures such as Vision Transformers and ConvNeXt, but similar accuracies for ResNets. Averaged over the datasets, SGD (Freeze-embed) is consistently the best or tied-best method for all 7 models (see avg. number on the right column).

		Liv-17	Wbirds	DomNet	FMoW	Camelyon	Avg.
CLIP ViT-B/16	SGD	80.0	62.5	69.6	37.3	86.8	67.2
CLIP ViT-B/16	AdamW	82.8	71.9	**89.2**	**40.7**	**95.7**	**76.0**
CLIP ViT-B/16	SGD (Freeze-embed)	**83.2**	**73.7**	88.2	40.2	94.3	**75.9**
Sup ViT-B/16	SGD	**89.5**	77.4	**86.3**	33.5	92.6	75.8
Sup ViT-B/16	AdamW	88.3	81.6	84.4	**35.9**	87.9	75.6
Sup ViT-B/16	SGD (Freeze-embed)	88.0	**82.4**	**86.3**	34.4	**93.7**	**77.0**
DINO ViT-B/16	SGD	**88.2**	56.1	76.0	33.6	86.9	68.2
DINO ViT-B/16	AdamW	87.4	61.2	77.4	**35.8**	**91.9**	70.7
DINO ViT-B/16	SGD (Freeze-embed)	86.7	**67.9**	**78.4**	**35.9**	90.6	**71.9**
BIT ResNet-50	SGD	**84.3**	**76.5**	80.0	34.1	90.4	73.1
BIT ResNet-50	AdamW	83.1	74.8	**84.0**	33.8	92.4	73.6
BIT ResNet-50	SGD (Freeze-embed)	84.1	75.5	82.3	**35.0**	**95.2**	**74.4**
BIT ResNet-101	SGD	82.8	76.9	**86.2**	**38.0**	89.3	74.6
BIT ResNet-101	AdamW	**82.9**	**79.4**	83.5	37.0	89.7	74.5
BIT ResNet-101	SGD (Freeze-embed)	**83.1**	77.3	86.0	36.0	**95.5**	**75.6**
ConvNext-Base	SGD	**94.0**	80.2	89.8	**39.7**	83.0	77.3
ConvNext-Base	AdamW	90.3	**89.8**	89.5	38.4	**89.5**	**79.5**
ConvNext-Base	SGD (Freeze-embed)	92.6	86.9	**91.2**	38.2	88.1	**79.4**
CLIP ViT-L/14	SGD	84.2	65.0	60.8	41.0	83.2	66.8
CLIP ViT-L/14	AdamW	88.0	**85.2**	**93.8**	48.3	95.9	82.2
CLIP ViT-L/14	SGD (Freeze-embed)	**90.5**	84.7	93.1	**49.9**	**96.5**	**83.0**

found surprising is that a minor tweak of freezing the embedding layer can make a large difference when fine-tuning with SGD, and achieve *comparable* or even slightly better accuracy than AdamW across a wide range of models and benchmarks.

More detailed results on CLIP. The CLIP models have particularly strong transfer learning performance and robustness, so we did a more detailed analysis of the CLIP ViT-B/16. [2] For SGD, AdamW, and SGD (Freeze-embed), we ran each hyperparameter configuration 3 times and computed 90% confidence intervals. SGD (Freeze-embed) achieves substantially better accuracies than SGD on all datasets, and comparable accuracies to AdamW both ID (Table 3) and OOD (Table 4).

SGD (Freeze-embed) and AdamW outperform LAMB and LARS. For the CLIP-ViT-B/16, we also ran two alternative adaptive gradient methods, LARS (You et al., 2017) and LAMB (You et al., 2020)—also sweeping over 6 learning rates, with a cosine learning rate schedule. One motivation for using LARS and LAMB is that they have a lower memory footprint than AdamW. While LARS and LAMB get higher accuracies than SGD, they do worse than SGD (Freeze-embed) and AdamW both ID and OOD. Averaged across the ID datasets, LARS (90.9%) and LAMB (91.7%) get slightly lower accuracies than SGD (Freeze-Embed) (92.1%) and AdamW (92.1%). Averaged across the OOD datasets, LARS (69.6%) and LAMB (73.2%) get much lower accuracies than SGD (Freeze-Embed) (75.9%) and AdamW (76.0%).

SGD (Freeze-embed) improves with scale. SGD (Freeze-embed) gets an 8.7% higher OOD accuracy than SGD on a CLIP-ViT-B/16 and similar OOD accuracy to AdamW. Scaling up to a large CLIP-ViT-L/14, SGD (Freeze-embed) gets a 14.3% higher OOD accuracy than SGD, and a 0.7%

[2] Generating Table 1 involved over 600 fine-tuning runs, so we weren't able to repeat this for every model.

Table 2: **In-distribution (ID)** accuracies for fine-tuning 7 popular models across 5 benchmark datasets, where we fine-tune with SGD, AdamW, or SGD (Freeze-Embed). SGD Freeze-embed consistly gets competitive accuracies with AdamW.

		Liv-17	Wbirds	DomNet	FMoW	Camelyon	Avg.
CLIP ViT-B/16	SGD	97.8	97.2	87.6	67.0	**99.4**	89.8
CLIP ViT-B/16	AdamW	**98.1**	**97.7**	**95.0**	**70.1**	**99.5**	**92.1**
CLIP ViT-B/16	SGD (Freeze-embed)	**98.2**	**97.8**	**94.9**	**70.0**	**99.5**	**92.1**
Sup ViT-B/16	SGD	**98.6**	**99.1**	**91.7**	64.1	**99.4**	90.6
Sup ViT-B/16	AdamW	**98.7**	**99.0**	**91.7**	**66.4**	**99.5**	**91.1**
Sup ViT-B/16	SGD (Freeze-embed)	**98.7**	**99.2**	91.5	65.0	**99.6**	90.8
DINO ViT-B/16	SGD	**98.4**	97.0	88.2	62.4	99.4	89.1
DINO ViT-B/16	AdamW	**98.5**	**97.9**	**89.4**	**66.0**	**99.6**	**90.3**
DINO ViT-B/16	SGD (Freeze-embed)	**98.4**	97.5	89.0	63.5	**99.5**	89.6
BIT ResNet-50	SGD	97.4	**98.4**	**89.3**	64.6	**99.5**	**89.8**
BIT ResNet-50	AdamW	97.2	**98.5**	**89.2**	**65.1**	**99.5**	**89.9**
BIT ResNet-50	SGD (Freeze-embed)	**97.6**	**98.5**	**89.2**	64.8	**99.5**	**89.9**
BIT ResNet-101	SGD	**98.3**	**98.9**	**92.0**	66.0	**99.4**	**90.9**
BIT ResNet-101	AdamW	**98.4**	98.6	91.1	**67.0**	**99.6**	**90.9**
BIT ResNet-101	SGD (Freeze-embed)	**98.4**	**98.8**	91.5	65.9	**99.5**	**90.8**
ConvNext-Base	SGD	**98.7**	99.0	94.8	66.3	99.4	91.6
ConvNext-Base	AdamW	**98.6**	**99.5**	94.5	**68.8**	**99.7**	**92.2**
ConvNext-Base	SGD (Freeze-embed)	**98.6**	**99.4**	**95.1**	67.4	**99.5**	**92.0**
CLIP ViT-L/14	SGD	98.4	97.3	84.3	69.0	**99.4**	89.7
CLIP ViT-L/14	AdamW	**98.9**	**98.8**	96.9	**74.5**	**99.6**	**93.7**
CLIP ViT-L/14	SGD (Freeze-embed)	**98.7**	**98.9**	**97.1**	**74.5**	**99.6**	**93.7**

Table 3: In-distribution (ID) accuracies for the CLIP base model, with 90% confidence intervals. SGD (freze-embed) is better than SGD, and competitive with AdamW, but LARS and LAMB perform worse than AdamW.

	Liv-17	Waterbirds	DomainNet	FMoW	Camelyon	Avg.
SGD	97.8 (0.2)	97.2 (0.1)	87.6 (6.4)	67.0 (0.8)	99.4 (0.0)	89.8
AdamW	98.1 (0.1)	97.7 (0.0)	95.0 (0.1)	**70.1 (0.2)**	**99.5 (0.0)**	92.1
SGD (Freeze-embed)	**98.2 (0.3)**	97.8 (0.1)	94.9 (0.3)	**70.0 (0.2)**	99.5 (0.0)	92.1
Gradual-unfreezing	**98.3 (0.1)**	**98.3 (0.0)**	**96.3 (0.1)**	69.2 (0.5)	99.3 (0.1)	**92.3**
LAMB	98.2	97.8	95.1	67.9	99.5	91.7
LARS	97.7	97.1	93.2	67.0	99.3	90.9

higher OOD accuracy than AdamW. This suggests that our findings might be even more relevant for larger models.

5 Gradual unfreezing and SoTA experiments

Gradual unfreezing. Instead of just freezing the first ("embedding") layer, we could start by freezing the entire pretrained model (except for the head) and gradually unfreeze the rest of the model over training. Strategies like this have been explored by ULMFiT (Howard & Ruder, 2018) and others (Mukherjee & Awadallah, 2019; Romero et al., 2020). An advantage of gradual unfreezing is that in earlier stages of training we tune a smaller fraction of the model, which speeds up training—for example, for WILDS-FMoW on an A100 GPU , the average runtime decreases by over 30% .

Table 4: Out-of-distribution (OOD) accuracies for the CLIP base model, with 90% confidence intervals. SGD (freze-embed) is much better than SGD, and competitive with AdamW, but LARS and LAMB performs much worse than AdamW.

	Liv-17	Waterbirds	DomainNet	FMoW	Camelyon	Avg.
SGD	80.0 (1.3)	62.5 (5.0)	69.6 (16.0)	37.3 (1.1)	86.8 (1.1)	67.2
AdamW	**82.8 (1.2)**	**71.9 (2.4)**	89.2 (1.1)	**40.7 (0.3)**	95.7 (0.4)	76.0
SGD (Freeze-embed)	**83.2 (0.8)**	**73.7 (1.1)**	88.2 (0.7)	**40.2 (0.7)**	94.3 (0.3)	75.9
Gradual-unfreezing	81.9 (0.1)	69.1 (3.4)	**93.2 (0.3)**	**40.5 (1.2)**	**96.5 (0.4)**	**76.2**
LAMB	79.5	64.0	90.4	38.8	93.4	73.2
LARS	**83.9**	48.6	83.8	38.6	93.3	69.6

Using a single learning rate throughout this process does not work, so we had to decay the learning rate exponentially (a cosine learning rate is not sufficient).

Results on CLIP-ViT-B/16. We show ID accuracies for gradual unfreezing in Table 3 and OOD accuracies in Table 4. While running faster, gradual unfreezing was able to get slightly better results both ID (92.3% vs. 92.1% for SGD Freeze-embed and 92.1% for AdamW) and OOD (76.2% vs. 75.9% for SGD Freeze-embed and 76.0% for AdamW). On some datasets, gradual unfreezing performed particularly well. For example, gradual unfreezing got an OOD accuracy of 96.5% on Camelyon—which is competitive with SGD (Freeze-embed) and AdamW on a much larger CLIP-ViT-L/14 model. Similarly, gradual unfreezing got an OOD accuracy of 93.2% on DomainNet which was over 4% better than AdamW and SGD (Freeze-embed) on a CLIP-ViT-B/16, and competitive with the two methods on the larger CLIP ViT-L/14.

However, we see that the majority of improvements come from simple freezing the embedding layer—which suggests that the embedding layer plays a particularly prominent role in modern vision architectures such as Vision Transformers and ConvNeXt.

SoTA results. Simply running SGD (Freeze-embed) on a CLIP-ViT-L/14 gives state-of-the-art results on 4/5 of the datasets: BREEDS-Living-17, DomainNet, WILDS-FMoW, and WILDS-Camelyon. Here, state-of-the-art means that the numbers we get are better than, to the best of our knowledge, any reported number and all numbers on the official leaderboard, and are better than standard full fine-tuning with SGD. For WaterBirds, AdamW on a ConvNeXt gets state-of-the-art accuracies. We show the best results in our paper in Table 5 with a comparison to the previous state-of-the-art, and we show results specifically for SGD (Freeze-embed) in Table 6.

To the best of our knowledge, the previous state-of-the-art results are from Wortsman et al. (2022) for WILDS-FMoW, Robey et al. (2021) for WILDS-Camelyon, Kumar et al. (2022) for BREEDS-Living-17 and the version of DomainNet introduced by Tan et al. (2020), and Ghosal et al. (2022) for Waterbirds. The WILDS numbers and references are taken from the official WILDS leaderboard.[3], and for Waterbirds we consider all methods that do not use group labels—for example, Liu et al. (2021) report 62.5% when no group labels are available. For Living-17, we omit all our models that use ImageNet data in pretraining because Living-17 is a subset of ImageNet.

6 Additional related works

Freezing parameters. Many works in transfer learning propose freezing parameters while fine-tuning to preserve pretrained information. For example, linear probing freezes the entire model except the head, and this can perform better out-of-distribution (Kumar et al., 2022; Wortsman et al., 2021). Using a model zero-shot without any fine-tuning can also improve out-of-distribution accuracies (Radford et al., 2021). In NLP, methods such as prefix-tuning (Li & Liang, 2021) and

[3]E.g., `https://wilds.stanford.edu/leaderboard/` in the case of WILDS, as of the ICLR deadline on September 28th.

Table 5: Our results compared with the best reported numbers in prior work on these datasets. We report out-of-distribution accuracies on each dataset—but all the methods *do not* use OOD data for hyperparameter selection or early stopping. For Waterbirds, we consider all methods that do not use group labels. Our numbers outperform prior work (cited in the main text) on all 5 datasets, although different datasets benefit from different optimization procedures.

	Liv-17	Waterbirds	DomainNet	FMoW	Camelyon
Best prior result	87.6	89.3	87.2	47.6	93.3
Our best result	**90.5**	**89.8**	**93.8**	**49.9**	**96.5**
Optimizer	SGD Freeze-Embed	AdamW	AdamW	SGD Freeze-Embed	SGD Freeze-Embed
Model	CLIP-ViT-L/14	ConvNeXt-B	CLIP-ViT-L/14	CLIP-ViT-L/14	CLIP-ViT-L/14

prompt-tuning (Lester et al., 2021) can improve out-of-distribution accuracy. Other works propose regularizing parameters towards initialization, using different learning rates for different layers, or tuning different layers for different examples (Ge & Yu, 2017; Guo et al., 2019; Zhang et al., 2020; Zhu et al., 2020; Jiang et al., 2021; Aghajanyan et al., 2021). Typically, a large fraction of the model is frozen to preserve the pretrained information—a key difference in this work is that we find that freezing a very small fraction of the model (e.g., 0.7% of the parameters in a vision transformer) can lead to substantial improvements in accuracy.

Fine-tuning optimizers. To address AdamW's larger memory footprint, many other deep learning optimizers have been proposed that use adaptive learning rates with a lower memory footprint, including LARS (You et al., 2017), LAMB (You et al., 2020), and AdaFactor (Shazeer & Stern, 2018).

7 Discussion.

We note that the methods we consider are not complex. The broader point of our work is that pretrained models (including CLIP Vision Transformers) have a lot of useful information from pretraining, but naively fine-tuning these models can lead to poor performance and does not leverage their capabilities. Exploring simple properties of these models such as the gradients of different layers, and designing simple modifications such as freezing just a small part of the model, can lead to state-of-the-art accuracies.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations (ICLR)*, 2021.

Peter Bandi, Oscar Geessink, Quirine Manson, Marcory Van Dijk, Maschenka Balkenhol, Meyke Hermsen, Babak Ehteshami Bejnordi, Byungjae Lee, Kyunghyun Paeng, Aoxiao Zhong, et al. From detection of individual metastases to classification of lymph node status at the patient level: the CAMELYON17 challenge. *IEEE Transactions on Medical Imaging*, 38(2):550–560, 2018.

Mathilde Caron, Hugo Touvron, Ishan Misra, Herve Jegou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *International Conference on Computer Vision (ICCV)*, 2021.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning (ICML)*, pp. 1597–1607, 2020.

Gordon Christie, Neil Fendley, James Wilson, and Ryan Mukherjee. Functional map of the world. In *Computer Vision and Pattern Recognition (CVPR)*, 2018.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Computer Vision and Pattern Recognition (CVPR)*, 2017.

Soumya Suvra Ghosal, Yifei Ming, and Yixuan Li. Are vision transformers robust to spurious correlations? *arXiv preprint arXiv:2008.02790*, 2022.

Boris Ginsburg, Patrice Castonguay, Oleksii Hrinchuk, Oleksii Kuchaiev, Vitaly Lavrukhin, Ryan Leary, Jason Li, Huyen Nguyen, Yang Zhang, and Jonathan M. Cohen. Training deep networks with stochastic gradient normalized by layerwise adaptive second moments. 2019.

Yunhui Guo, Honghui Shi, Abhishek Kumar, Kristen Grauman, Tajana Rosing, and Rogerio Feris. Spottune: Transfer learning through adaptive fine-tuning. In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2016.

Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Association for Computational Linguistics (ACL)*, 2018.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. In *International Conference on Learning Representations (ICLR)*, 2021.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning (ICML)*, 2021.

Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *ECCV*, 2020.

Simon Kornblith, Jonathon Shlens, and Quoc V. Le. Do better imagenet models transfer better? In *Computer Vision and Pattern Recognition (CVPR)*, 2019.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. In *International Conference on Learning Representations (ICLR)*, 2022.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Association for Computational Linguistics (ACL)*, 2021.

Evan Zheran Liu, Behzad Haghgoo, Annie S. Chen, Aditi Raghunathan, Pang Wei Koh, Shiori Sagawa, Percy Liang, and Chelsea Finn. Just train twice: Improving group robustness without training group information. In *International Conference on Machine Learning (ICML)*, 2021.

Zhuang Liu, Hanzi Mao, Chaozheng Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Computer Vision and Pattern Recognition (CVPR)*, 2022.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

Subhabrata Mukherjee and Ahmed Hassan Awadallah. Distilling transformers into simple neural networks with unlabeled transfer data. *arXiv preprint arXiv:1910.01769*, 1, 2019.

Xingchao Peng, Qinxun Bai, Xide Xia, Zijun Huang, Kate Saenko, and Bo Wang. Moment matching for multi-source domain adaptation. In *International Conference on Computer Vision (ICCV)*, 2019.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, volume 139, pp. 8748–8763, 2021.

Alexander Robey, George J. Pappas, and Hamed Hassani. Model-based domain generalization. In *NeurIPS*, 2021.

Miguel Romero, Yannet Interian, Timothy Solberg, and Gilmer Valdes. Targeted transfer learning to improve performance in small medical physics datasets. *Medical Physics*, 47(12):6246–6256, 2020.

Shiori Sagawa, Pang Wei Koh, Tatsunori B. Hashimoto, and Percy Liang. Distributionally robust neural networks for group shifts: On the importance of regularization for worst-case generalization. In *International Conference on Learning Representations (ICLR)*, 2020.

Shibani Santurkar, Dimitris Tsipras, and Aleksander Madry. Breeds: Benchmarks for subpopulation shift. *arXiv*, 2020.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning (ICML)*, 2018.

Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *ArXiv*, abs/2106.10270, 2021.

Shuhan Tan, Xingchao Peng, and Kate Saenko. Class-imbalanced domain adaptation: An empirical odyssey. *arXiv preprint arXiv:1910.10320*, 2020.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herv'e J'egou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021.

Mitchell Wortsman, Gabriel Ilharco, Mike Li, Jong Wook Kim, Hannaneh Hajishirzi, Ali Farhadi, Hongseok Namkoong, and Ludwig Schmidt. Robust fine-tuning of zero-shot models. *arXiv preprint arXiv:2109.01903*, 2021.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning (ICML)*, 2022.

Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv: Computer Vision and Pattern Recognition*, 2017.

Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations (ICLR)*, 2020.

Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv*, 2020.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: A baseline for network adaptation via additive side networks. In *European Conference on Computer Vision (ECCV)*, 2020.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. FreeLB: Enhanced adversarial training for natural language understanding. In *International Conference on Learning Representations (ICLR)*, 2020.

Table 6: Our best result using SGD (Freeze-embed) outperforms previous state-of-the-art in 4/5 datasets—all datasets except Waterbirds. The difference with Table 5 is that Table 5 shows our best results across models and optimizers, and here we show the best results for SGD (Freeze-embed).

	Liv-17	Waterbirds	DomainNet	FMoW	Camelyon
Best prior result	87.6	**89.3**	87.2	47.6	93.3
Our best result	**90.5**	86.9	**93.1**	**49.9**	**96.5**
Optimizer	SGD Freeze-Embed	SGD Freeze-Embed	SGD Freeze-Embed	SGD Freeze-Embed	SGD Freeze-Embed
Model	CLIP-ViT-L/14	ConvNeXt-B	CLIP-ViT-L/14	CLIP-ViT-L/14	CLIP-ViT-L/14

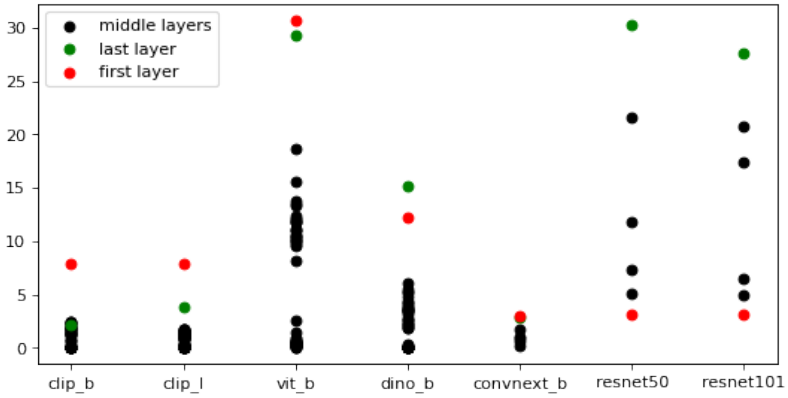


Figure 3: We visualize the norm of the gradients at different layers of each neural network, on Living-17. The first ("embedding") layer is shown in red and an outlier in all cases, but is *smaller* than other layers for ResNet models, and *larger* for the Vision Transformers and ConvNeXt. This plot is similar to Figure 2 but for Living-17.

A Additional Plots and Results

In Table 6 we show the best results in our paper when using SGD (Freeze-embed).

In Figure 3 and Figure 4 we show plots for the gradients at different layers for Living-17 and Waterbirds respectively.

Additional training details. We sweep over 6 learning rates for SGD ([3e-5, 1e-4, 3e-4, 1e-3, 3e-3, 1e-2]) and for AdamW ([3e-7, 1e-6, 3e-6, 1e-5, 3e-5, 1e-4]). As standard, we use smaller learning rates for AdamW because they work better. We check that the best learning rate is in the middle of this grid. For SGD we use momentum of 0.9, and besides that use all the default options for SGD and AdamW in PyTorch.

We train for 20 epochs on Living-17, 20 epochs on Waterbirds, 50 epochs on DomainNet, 5 epochs on WILDS-FMoW, and 3 epochs on Camelyon.

We use very basic augmentations (e.g., only horizontal flips for WILDS, and standard augmentations from past work), include for our state-of-the-art results:

1. For WILDS-FMoW, the images are 224x224, so we do not resize, and only perform a random horizontal flip augmentation.

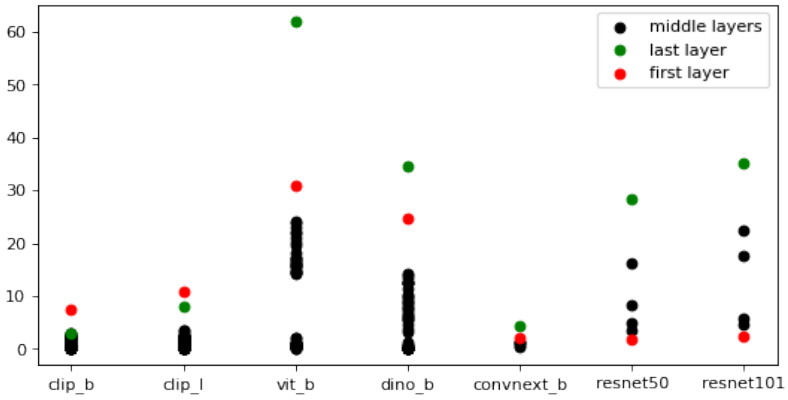


Figure 4: We visualize the norm of the gradients at different layers of each neural network, on Waterbirds. The first ("embedding") layer is shown in red and an outlier in all cases, but is *smaller* than other layers for ResNet models, and *larger* for the Vision Transformers and ConvNeXt. This plot is similar to Figure 2 but for Waterbirds.

2. For WILDS-Camelyon, we resize the images to 224x224 with a bilinear interpolation (standard in Torch), and only perform a random horizontal flip augmentation. We do not perform any augmentations at test time.
3. For BREEDS-Living-17, we perform a RandomResizedCrop to 224x224 sized images (using the default options in Torch), and then a random horizontal flip augmentation. We do not perform any augmentations at test time.
4. For BREEDS-Living-17, we follow (Kumar et al., 2022) and perform a RandomResizedCrop to 224x224 sized images (using the default options in Torch), and then a random horizontal flip augmentation while training. At test-time we resize the image to 256 and then take a centercrop of size 224x224.
5. For DomainNet, we follow (Kumar et al., 2022) and first resize the image to 256x256 with bicubic interpolation, and then take a RandomCrop of size 224 (using the default options in Torch), and then a random horizontal flip augmentation while training. At test-time we resize the image to 224x224 with bicubic interpolation.
6. For Waterbirds, we resize the image to 224 with bicubic interpolation and then take a centercrop of size 224. At test-time we resize the image to 224x224 with bicubic interpolation. We apply the same transformation at test time.