A Simple Finetuning Strategy Based on Bias-Variance Ratios of Layer-Wise Gradients

Mao Tomita[1], Ikuro Sato[2,1], Rei Kawakami[1], Nakamasa Inoue[1], Satoshi Ikehata[3,1], and Masayuki Tanaka[1]

[1] Institute of Science Tokyo, Tokyo, Japan
[2] Denso IT Laboratory, Tokyo, Japan
[3] National Institute of Informatics, Tokyo, Japan

Abstract. Finetuning is an effective method for adapting pretrained networks to downstream tasks. However, the success of finetuning depends heavily on the selection of layers to be tuned, as full finetuning can lead to overfitting, while tuning only the last layer may not capture the necessary task-specific features. This requires a balanced approach of automatic layer selection to achieve higher performance. In this context, we propose the Bias-Variance Guided Layer Selection (BVG-LS), a simple yet effective strategy that adaptively selects a layer to be tuned at each training iteration. More specifically, BVG-LS computes the bias-variance ratios of mini-batch gradients for each layer and updates the parameters of the layer with the largest ratio. This strategy reduces the risk of overfitting while maintaining the model's capacity to learn task-specific features. In our experiments, we demonstrate the effectiveness of the BVG-LS strategy on seven image classification tasks. We show that BVG-LS outperforms full finetuning on all tasks with the WideResNet-50-2 model and on six out of seven tasks with the ViT-S model.[4]

Keywords: Bias-variance tradeoff · Finetuning strategy · Image classification

1 Introduction

Finetuning is the process of adapting a pretrained model to a new, but related, target task [20, 47]. This approach can leverage the general knowledge captured during pretraining on the source task. It also saves time and computational resources compared to training a new model from scratch. Finetuning is especially useful when a limited amount of training data is available for the target task [46].

Two typical strategies of finetuning are linear probing and full finetuning [56]. Linear probing only updates the last layer of the model (*i.e.*, the linear classification layer) as shown in Fig. 1 (a). This is an efficient strategy because the layers for feature extraction are frozen. However, it might not adapt well to the target task. In contrast, full finetuning updates all layers as shown in Fig. 1 (b).

[4] The reproduction code is available online.
http://www.vip.sc.e.titech.ac.jp/proj/BVGLS/

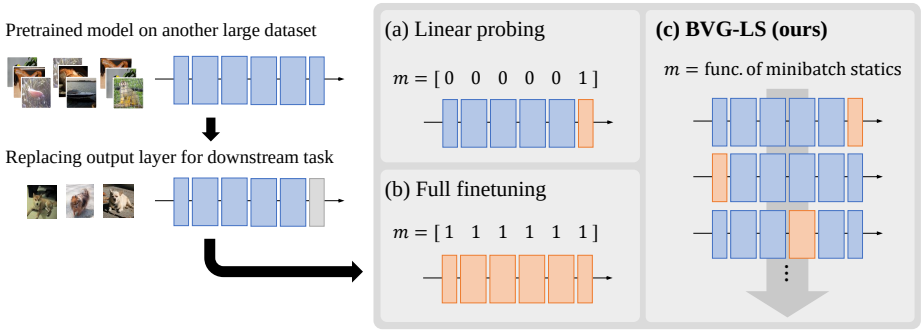


Fig. 1: Summary of finetuning strategies. (a) Linear probing updates only the last layer. (b) Full finetuning updates the all layers. (c) Our BVG-LS strategy adaptively selects a layer to be updated at each training iteration. The binary mask m indicates layers to be updated. BVG-LS determines m from the bias-variance ratios of layer-wise mini-batch gradients.

This is effective in the sense that it allows the model to learn task-specific features, potentially leading to better performance in the target task. However, it comes with several drawbacks, including the risk of overfitting, loss of robustness [22, 51], and high computational costs.

Based on the observation that some layers can extract common general features across tasks, while others are more specific to a particular task [53, 55], freezing or updating only a subset of layers can be more effective than using linear probing or full finetuning [45, 53]. As such, layer selection is an important factor that affects the performance in the target task. However, it is not trivial to determine how many and which layers should be updated, as the appropriate criteria depend on the relationship between the source and target tasks.

Another important factor in fine-tuning is the adaptive selection of layers to be tuned during training. Gradually tuning from the last layer to the first layer [16] or automatically selecting layers to be tuned for each sample [12] are shown to provide performance improvement. This might not only be due to better initialization in learning but also, as some previous studies suggest, due to the prevention of co-adaptation between layers, which can positively impact generalizability [14, 41, 42, 53].

How can we select the best layers to tune? To address this question, we collect the statistics of the mini-batch gradients. The bias (the average) of the gradients will be dominant when training should continue, while the variance of the gradients will be larger at a stationary point such as the minimum. Additionally, the flatness of the minimum is an important index for generalization performance [10, 21, 42]. At a flat minimum point, we can assume that the variance of the mini-batch gradients is large while the bias is close to zero. Thus, the bias-variance ratio is a good measure to determine whether the parameters should be further updated. Our method collects the bias and variance statistics by layers, as this simplifies computation and can break inter-layer co-adaptation.

Based on those assumptions, this paper introduces a simple yet effective strategy, namely the Bias-Variance Guided Layer Selection (BVG-LS) strategy. More specifically, BVG-LS measures the bias-variance ratios of mini-batch gradients for each layer at each training iteration and adaptively selects the layer with the maximum ratio to be tuned. This strategy enables models to learn task-specific features while avoiding overfitting because the bias-variance ratios can be understood as an indicator of the risk of overfitting. It is worth noting that this strategy can be applied to any target tasks and models, as the proposed layer selection method is independent of the tasks and models.

The main contributions of this paper are as follows:

- We propose BVG-LS, a simple finetuning strategy that adaptively selects layers to be tuned, based on the bias-variance ratios of layer-wise gradients.
- We conducted experiments on seven target tasks with multiple models. The results show that our method outperforms full fine-tuning on all tasks for WideResNet-50-2 and on six tasks for ViT-S.

2 Related Work

Finetuning is an effective training method for various downstream tasks. Early studies find the power of features pretrained on ImageNet [39] in object detection [8, 11, 43] and fine-grained classification [29]. Kornblith *et al.* [24] finds transferability of the pretrained model is highly correlated with ImageNet-top-1 accuracy when pretrained on ImageNet, and Kolesnikov *et al.* [23] shows the size of the pretraining dataset also matters. To achieve high accuracy in downstream tasks, many approaches aim to improve pretraining methods [18,34,40], but this paper focuses on the learning method during finetuning.

Some studies show that linear probing suffices [1, 6], while many others indicate that full finetuning yields better results [5, 19, 24, 31, 36, 40]. Thus, there must be a compromise between them that can adaptively select which layers and parameters to be updated based on the relationship between source and target tasks. In fact, Yosinski *et al.* [53] shows that the layers closer to the output are more task-specific, thus need to be re-trained for new tasks. Azizpour *et al.* shows that if the distance between the source and target tasks is large (small), the features from early (later) layers should be re-trained [3]. Long *et al.* [32] shows the use of 10 times larger learning rate for the classification layer yields good transfer.

Based on those findings, many finetuning methods are proposed, focusing on which layers and parameters to be updated during finetuning. They can be categorized into two groups. The first group selects the layers or parameters first, and fixes them during finetuning. Shen *et al.* [45] utilizes evolutionary search to select layers, while the search algorithm requires validation data. Howard *et al.* [16] and Romero *et al.* [38] manually set layer-wise learning rate, where the learning rate decreases linearly from the output to input layers. (They [16, 38] also introduce scheduling that gradually unfreezes layers from the output layer during training.) Similar to this, Kumar *et al.* [26] splits the learning phase into

two, tuning only the final layer in the former and all layers in the latter. Xu *et al*. [50] and Zhang *et al*. [57] mask certain layers or parameters in advance, based on the Fisher information of the loss and norm of the gradient, obtained from partial samples, respectively.

The second group adaptively selects which layers to be tuned during training. Lee *et al*. [27] sets larger learning rate for layers with larger gradient norms. The learning rates are refreshed at each epoch. Guo *et al*. [12] selects the layers by having another network that learns the policy about this selection. Our method selects layers at each iteration; thus, the learning will be more flexible. It also has relatively less computational cost compared to the calculation of Fisher information matrix or having another network for layer selection.

There are other lines of work in finetuning, such as those that consider regularization in features [52], interpolation of weights between pretrained and finetuned models [48] such as in model merging [2], and minimizing structural risk considering model complexity [44]. These are orthogonal to our method; thus, incorporating these techniques may improve our results. Parameter-efficient tuning such as adapters [15], LoRA [17], and prompt tuning [28] are also intensively studied for transferring the knowledge of large models. They are also different lines of work, but it may be possible to incorporate our bias-variance-based selection into these learning.

3 Method

3.1 Overview

We present BVG-LS, a simple yet effective finetuning strategy that adaptively selects the layer to be tuned at each training iteration based on the bias-variance ratios of layer-wise gradients.

Notation and settings. Let $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$ be a training dataset, where $\mathcal{X}$ is a set of images and $\mathcal{Y}$ is a set of labels. This paper considers supervised finetuning, where the goal is to train a function $f : \mathcal{X} \to \mathcal{Y}$ that accurately predicts labels from images. We denote by $\hat{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$ the predicted label, where $\boldsymbol{x} \in \mathcal{X}$ is an image and $\boldsymbol{\theta}$ is a set of learnable parameters. To discuss the choice of layers to be tuned, we assume that each layer has independent parameter subset, *i.e.*, $\boldsymbol{\theta} = (\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \ldots, \boldsymbol{\theta}^{(L)})$ where $\boldsymbol{\theta}^{(l)}$ is the subset of parameters for the l-th layer and L is the number of layers.

Update rule. Let $\mathcal{B} \subset \mathcal{D}$ denote a mini-batch randomly sampled from training data. We can generally define the parameter update rule as follows:

$$\boldsymbol{\theta}^{(l,t)} \leftarrow \boldsymbol{\theta}^{(l,t-1)} + m^{(l,t)}\, \boldsymbol{d}^{(l,t)}, \tag{1}$$

where $t \in \mathbb{N}$ is the index of training iteration, $\boldsymbol{\theta}^{(l,t)}$ is the subset of parameters for the l-th layer at the t-th iteration, $\boldsymbol{d}^{(l,t)}$ is the update direction, and

$m^{(l,t)} \in \{0,1\}$ is a *tuning mask* to determine which layer to be tuned. The update direction is given by $\boldsymbol{d}^{(l,t)} = -\eta\, \boldsymbol{g}^{(l,t)}$ where $\eta \in \mathbb{R}$ is a learning rate and $\boldsymbol{g}^{(l,t)} = \nabla_{\boldsymbol{\theta}^{(l)}} \mathcal{L}$ is the mini-batch gradient computed with a pre-defined loss function $\mathcal{L}$ and the mini-batch $\mathcal{B}$, for SGD. It can be easily extended to the momentum SGD and ADAM. The l-th layer is updated if and only if $m^{(l,t)} = 1$. The BVG-LS strategy determines $m^{(l,t)}$ based on bias-variance ratios of layer-wise gradients. Note that setting $m^{(L,t)} = 1$ and $m^{(l,t)} = 0\,(l \neq L)$ for all t reduces to linear probing, and setting $m^{(l,t)} = 1$ for all l and t reduces to full finetuning.

3.2 The BVG-LS strategy

At each training iteration, the BVG-LS strategy selects a layer to be tuned based on the bias-variance ratios of layer-wise mini-batch gradients. More specifically, it computes the bias B[5] and the variance V with respect to the mini-batch gradients, and selects the layer with the largest bias-variance ratio $r = B/V$. Here, the bias B is defined by the squared magnitude of the expected value of mini-batch gradients *i.e.*, $B = \|\mathbb{E}[\boldsymbol{g}^{(l,t)}]\|^2$, and the variance V is defined by the trace of the covariance of mini-batch gradients, *i.e.*, $V = \mathrm{tr}(\mathbb{V}[\boldsymbol{g}^{(l,t)}])$.

Figure 2 provides an intuitive understanding of why the BVG-LS strategy works and reduces the risk of overfitting. When the bias is larger compared to the variance, many mini-batch gradients share the same parameter update direction, as shown in Fig. 2 (a). In this case, updating parameters helps decrease the global loss. Conversely, when the bias is smaller compared to the variance, the update directions are likely to vary, as shown in Fig. 2;(b). In this case, the risk of overfitting to specific mini-batches increases, and thus, skipping parameter updates reduces the risk of overfitting. Therefore, the BVG-LS strategy uses the bias-variance ratio to determine which layer to be updated.

In practice, calculating the bias and variance at each iteration is computationally expensive because it requires to evaluate mini-batch gradients for a number of mini-batches to estimate the expected value of the mini-batch gradient. To address this problem, the BVG-LS strategy uses the exponential moving averages (EMAs). Specifically, the bias-variance ratio is computed by

$$r^{(l,t)} = \frac{\sum_i \left(b_i^{(l,t)}\right)^2}{\sum_i \left(v_i^{(l,t)} - \left(b_i^{(l,t)}\right)^2\right)}, \tag{2}$$

where $b_i^{(l,t)}$ and $v_i^{(l,t)}$ are EMAs of mini-batch gradients and the squared ones, respectively, given by

$$b_i^{(l,t)} = \alpha b_i^{(l,t-1)} + (1-\alpha) g_i^{(l,t)}, \tag{3}$$

$$v_i^{(l,t)} = \alpha v_i^{(l,t-1)} + (1-\alpha)\left(g_i^{(l,t)}\right)^2. \tag{4}$$

[5] Strictly speaking, B is the square of bias, but we refer to B as bias for simplicity.

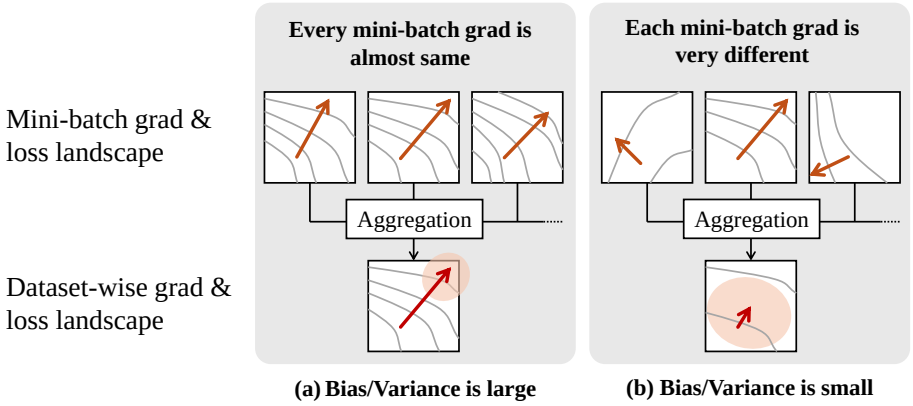


Fig. 2: Relationship between the learning state and the bias/variance of the mini-batch gradients. The gray curve represents the loss landscape, the orange arrow represents the mini-batch gradient, the red arrow represents the mean of the mini-batch gradient (*i.e.*, data-wise gradient), and the red area represents the variance of the mini-batch gradient.

Here, $g_i^{(l,t)}$ is the i-th element of $\boldsymbol{g}^{(l,t)}$ and $0 \leq \alpha \leq 1$ is a momentum hyperparameter. It is worth noting that $r^{(l,t)}$ is a dimensionless quantity, and is independent of the dimension and scale of the parameters. This indicates that $r^{(l,t)}$ can be compared between layers.

Finally, we define the tuning mask so that only the layer with the largest value of $r^{(l,t)}$ is updated at each iteration. The definition of the tuning mask is given by

$$m^{(l,t)} = \begin{cases} 1 & \text{if } l = \arg\max_{l'} r^{(l',t)} \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

Note that as for the size of mini-batch, our method does not impose any restrictions. Use of large mini-batches gives more accurate bias estimates $b_i^{(l,t)}$, whereas the scale of the variance estimates $v_i^{(l,t)}$ is obviously affected by the size of mini-batches. However, this scale ambiguity does not essentially affect the order of the variance estimates across layers, ignoring the statistical noises. Thus, it can be said that our layer selection mechanism is generally robust to the different mini-batch sizes.

Algorithm 1 summarizes the BVG-LS strategy. It iterates through the following five steps. First, the mini-batch gradients $\boldsymbol{g}^{(l)}$ for each layer are computed using a mini-batch $\mathcal{B}$ drawn from the training dataset $\mathcal{D}$. Second, the bias and variance is updated as Eqs. (3) and (4), respectively. Third, the bias-variance ratio is computed by Eq. (2). Finally, only the parameters of the layer with the largest ratio are updated. Note that the EMAs of bias and variance are updated regardless of whether the parameters are updated.

Algorithm 1 BVG-LS

Require: initial parameter set $\{\boldsymbol{\theta}^{(l)}\}_{l=1}^{L}$, training dataset $\mathcal{D}$, loss function $\mathcal{L}$

$b_i^{(l)} \leftarrow 0$, $v_i^{(l)} \leftarrow 1\ (\forall\, l, i)$
for $t = 1 : T$ **do**
  Draw a mini-batch $\mathcal{B}$ from $\mathcal{D}$.
  for $l = 1 : L$ **do**
    $\boldsymbol{g}^{(l)} \leftarrow \nabla_{\boldsymbol{\theta}^{(l)}} \mathcal{L}$
    $b_i^{(l)} \leftarrow \alpha b_i^{(l)} + (1-\alpha) g_i^{(l)}\ (\forall i)$
    $v_i^{(l)} \leftarrow \alpha v_i^{(l)} + (1-\alpha)(g_i^{(l)})^2\ (\forall i)$
    $r^{(l)} \leftarrow (\sum_i (b_i^{(l)})^2)/(\sum_i (v_i^{(l)} - (b_i^{(l)})^2))$
  end for
  $l^* \leftarrow \arg\max_l r^{(l,t)}$
  $\boldsymbol{\theta}^{(l^*)} \leftarrow \boldsymbol{\theta}^{(l^*)} - \eta \boldsymbol{g}^{(l^*)}$
end for

Table 1: Summary of source and target datasets. The numbers of classes, training images, and test images are provided. ImageNet-1K is used as the source dataset and the others are used as target datasets.

Dataset	# of classes	# of training images	# of test images
ImageNet-1K [39]	1,000	1,281,167	50,000
Flowers [35]	102	1,020	1,020
Pets [37]	37	3,680	3,669
DTD [7]	47	1,880	1,880
Aircraft [33]	30	3,334	3,333
Food [4]	101	75,750	25,250
SUN [49]	397	54,377	54,377
CIFAR100 [25]	100	50,000	10,000

4 Experiments

In this section, we demonstrate the effectiveness of our BVG-LS method by comparing with full finetuning, partial finetuning and linear probing on seven image classification datasets with multiple model architectures.

4.1 Experimental settings

Datasets. In all the experiments, the ImageNet-1K dataset [39] was used as the source dataset to pretrain common models, which were then finetuned by different methods. As target datasets, we chose seven image classification datasets: Oxford 102 Flowers [35], Oxford-IIIT Pets [37], Describable Textures Dataset (DTD) [7], FGVC Aircraft [33], Food-101 [4], SUN397 [49] and CIFAR100 [25]. The numbers of classes and images contained in these datasets are summarized in Tab. 1.

Table 2: Model sizes (the numbers of layers and parameters) and ImageNet-1K error rates of pretrained models used in the experiments.

Model	# of layers	# of parameters	Top-1 error rate
Wide-ResNet-50-2 [54]	6	68.9 M	20.35 %
ViT-S [9]	14	22.1 M	21.15 %
ViT-B [9]	14	86.6 M	19.76 %
ConvNeXt-T [30]	6	28.6 M	17.30 %

Model architectures. Wide-ResNet-50-2 [54], ViT-S, ViT-B [9], and ConvNeXt-T [30] were used for pretraining and finetuning. Wide-ResNet, convolutional architecture with many channels, was designed as an improved version of ResNet [13]. Wide-ResNet-50-2 consists of an input layer, 4 hidden layers, and an output layer, making a 6-layered structure[6]. Vision transformer architectures ViT-S and ViT-B both consist of an input layer, 12 Transformer encoder layers, and an output layer, having total of 14 layers. ConvNeXt-T is more recent convolutional architecture that consists of an input layer, 4 stages, and an output layer, having total of 6 layers. The numbers of layers, the numbers of parameters, and the error rates of pretrained models on the ImageNet-1K validation set are summarized in Tab. 2.

Training details. After pretraining, the weight matrix in the output layer of each pretrained model is replaced by a random matrix that can produce a vectors of dimension same as the number of classes in the target dataset for finetuning. ViT-S, ViT-B, and ConvNeXt-T were finetuned for 300 epochs, whereas Wide-ResNet-50-2 was finetuned for 50 epochs on each target dataset. We employed Nesterov's accelerated gradient method with momentum rate of 0.9 in all experiments. The batch size and learning rate were 128 and 0.01, respectively. No learning rate scheduling was adopted in the finetuning stage. The coefficient used in the EMA computations of biases and variances, α, appeared in Eqs. (3) and (4), was set to 0.9.

4.2 Results

Generalization. We first observe funetuning performance of different methods using the strong convolutional model, WideResNet-50-2. Tab. 3 provides the summary of test error rates. Notably, full finetuning (or end-to-end finetuning) and linear probing (updating only the final layer) perform poorly, compared to other methods, even though these two naive strategies are widely adopted. Partial finetuning performs better than full finetuning and linear probing in most cases, but still the test error rates fluctuate significantly across different settings of partial finetuning, depending on which layers are frozen. This indicates that

[6] Here, "layer" refers to a group of similar operations (e.g., convolution).

Table 3: Test error rates (%) after finetuning the WideResNet-50-2 model pretrained on ImageNet-1K. "Ln-m" indicates that the n-th layer, $n+1$-layer, $\cdots$, and m-th layer are finetuned jointly from pretrained values, while the rest of the layers are kept fixed. Note that 6-th layer is the final layer. "Linear probing" indicates that only the final layer is finetuned. "Full finetuning" is same as L1-6 for this model. The best and second-best results are marked in bold and underlined, respectively.

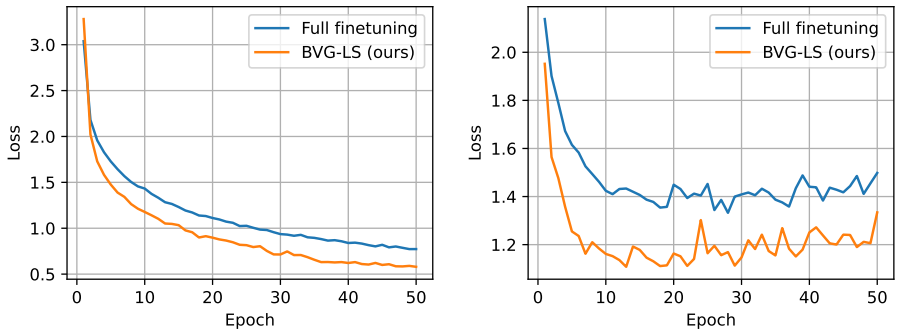
Dataset	Full finetuning	Partial finetuning				Linear probing	BVG-LS (ours)
		L2-6	L3-6	L4-6	L5-6		
Flowers	7.65	6.67	6.37	<u>5.88</u>	**4.61**	12.84	6.18
Pets	15.14	13.24	11.96	10.29	8.33	<u>8.22</u>	**6.26**
DTD	40.37	38.78	37.23	34.57	<u>31.60</u>	35.32	**29.20**
Aircraft	19.34	18.32	16.28	<u>16.22</u>	17.72	49.40	**14.60**
Food	18.01	17.05	16.52	<u>15.84</u>	16.97	35.45	**14.19**
SUN	35.54	35.18	34.23	33.21	<u>31.77</u>	36.72	**30.23**
CIFAR100	22.07	<u>21.65</u>	21.89	23.06	31.93	56.87	**17.25**
Average	22.59	21.55	20.64	<u>19.87</u>	20.42	33.55	**16.84**

practitioners need to conduct computation-intensive hyperparameter search for finetuning on each dataset to enhance the model's generalization ability. Our BVG-LS outperforms full finetuning, partial finetuning and linear probing on Pets, DTD, Aircraft, Food, SUN and CIFAR100 datasets. The only exception is Flower dataset, where BVG-LS marks the top-3 performance.

It is worth mentioning that our BVG-LS significantly outperforms full finetuning, although these two methods do update parameters of all layers in the end. As will be shown later, we have evidence that BVG-LS updates all layers during finetuning. This observation indicates that the finetuning scheme of coherently updating all layers likely deteriorates generalization abilities of a model. In addition, given that BVG-LS selects a layer to be updated at a time, full finetuning effectively experiences 6 times more updates in an arbitrary layer, on the average. Nevertheless, the learning curve in Fig. 3 shows that BVG-LS undergoes faster descent than the full finetuning in terms of both empirical and test losses. We confirmed similar trends on other datasets. Updating only one layer at a time might sound ineffective because computed gradients are partly "thrown out"[7], but in reality this sparse update scheme enjoys faster loss descent, while avoiding excessive interference with strong feature extraction ability of the pretrained model.

Next, we observe the test performance of ViT-S, ViT-B and ConvNext/T as summarized in Tab. 4. Among the methods compared, linear probing consistently underperforms the others except for only a few cases. In the case of ViT-S, out BVG-LS clearly outperforms full finetuning on all datasets except for Aircraft. However, with ViT-B, which has about 4 times more parameters compared to ViT-S, BVG-LS won 4 out of 7 datasets and underperforms on

[7] We do use all gradients to update the statistical measures, though.



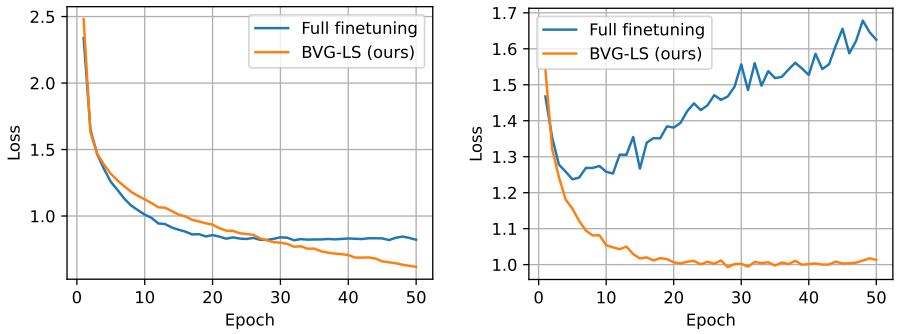
(a) Training loss **(b)** Test loss

Fig. 3: Comparison of loss curves between BVG-LS and full finetuning: (a) loss on training data, (b) loss on test data. Results of the WideResNet-50-2 models finetuned on the SUN dataset are shown. Our BVG-LS undergoes faster loss descent in both training and test cases, and achieves smaller test loss than full finetuning.

Table 4: Performance comparison using ViT-S, ViT-B, and ConvNeXt-T. Full: full finetuning. Linear: linear probing. The best results are marked in bold.

Dataset	ViT-S			ViT-B			ConvNeXt-T		
	Full	Linear	BVG-LS	Full	Linear	BVG-LS	Full	Linear	BVG-LS
Flowers	5.78	10.39	**5.39**	**4.61**	7.16	6.86	**3.53**	15.20	5.39
Pets	8.88	7.84	**6.32**	5.09	5.31	**4.98**	**5.12**	5.94	5.20
DTD	30.69	32.77	**25.75**	**25.85**	28.46	26.60	25.48	30.05	**25.05**
Aircraft	**17.93**	46.28	18.47	**13.28**	33.39	18.47	**8.84**	38.34	11.42
Food	17.88	28.34	**13.24**	12.66	20.20	**11.96**	13.71	23.58	**12.51**
SUN	33.93	32.83	**28.05**	27.58	28.94	**26.37**	27.93	30.21	**26.71**
CIFAR100	17.31	22.40	**12.39**	10.25	15.82	**10.10**	13.90	22.25	**12.06**
AVE	18.91	25.84	**15.66**	**14.19**	19.90	15.05	14.07	23.65	**14.05**

the average test error rate compared to full finetuning. With the ConvNeXt-T model, a convolutional model having 2.4× less parameters than WideResNet-50-2, BVG-LS again won 4 out of 7 datasets and performs equally on the average test error rate compared to full finetuning. These observation brings implications that BVG-LS performs mostly better than other schemes compared when relatively large-size convolutional models (e.g., WideResNet) or realtively small-size transformer models (e.g., ViT-S) are used. Currently, we have no clear explanation for these implications. Nevertheless, in the opposite cases (e.g., ViT-B or ConvNeXt-T), model performace depends on datasets; therefore, in reality trying both full finetuning and BVG-LS is a good practice. To provide an example of how BVG-LS undergoes loss descent in a consistent fashion, Fig. 4 shows loss curves of BVG-LS and full finetuning on the SUN dataset. It is clear that full



(a) Train loss **(b)** Test loss

Fig. 4: Comparison of loss curves between BVG-LS and full fine-tuning: (a) loss on training data, (b) loss on test data. Results of the ViT-S models finetuned on the SUN dataset are shown. Our BVG-LS achieves much smaller test loss than full finetuning.

finetuning quickly altered to overfitting stage before reaching a sufficiently small test loss.

Analyses on layer selection frequencies in BVG-LS. We investigated the frequency for each layer to be updated in BVG-LS finetuning. The proportion of layer selections during BVG-LS finetuning on each dataset is plotted in Fig. 5. Layer selection frequencies exhibit great fluctuations in the cases of Flowers, Pets, DTD and Aircraft datasets, due to the fact that they have less training data than the others (see Tab. 1), resulting in small numbers of updates per epoch. As stated earlier, in all cases, every layer is updated, even though the update frequencies vary across different layers. Among 6 layers, the last layer undergoes dominant updates in the very early stage of finetuning on all target datasets, except Flowers and Aircraft. It is continued to be relatively high throughout the entire finetuning. This makes sense, since the last layer, which is randomly initialized for a target dataset, likely yield higher bias-variance gradient ratios than the other layers. Somewhat surprisingly, the selection frequency of the input layer is quite high (roughly, the second highest). This may be explained by the differences in the low-level features of the source and target datasets.

Analyses of layer-wise bias-variance ratios. We analyze the layer-wise bias-variance ratios $r^{(l,t)}$ defined in Eq. (2). Fig. 6 shows $r^{(l,t)}$ computed during full finetuning and BVG-LS finetuning on each dataset. As explained earlier, we assumes that the smaller this bias-variance ratio is for a layer, the more likely it is to overfit.

First, let us observe the full finetuning results in first and third columns of Fig. 6. The bias-variance ratios have considerable disparities across layers. For example, in Pets, DTD, and Food, the bias-variance ratios for layer 1 (input layer) and layer 6 (output layer) are significantly smaller compared to other

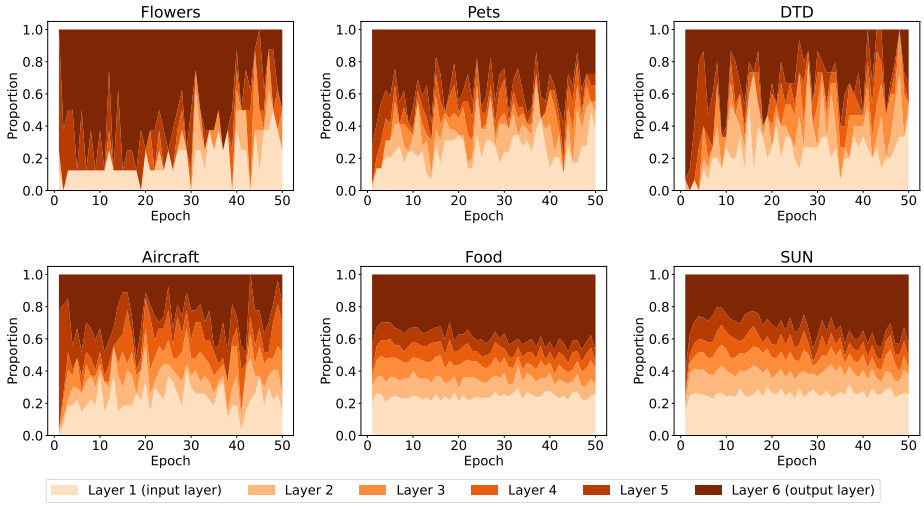


Fig. 5: Layer selection frequencies in the BVG-LS finetuning. Results of WideResNet-50-2 models are shown. All 6 layers are updated in the finetuning process. The layer 6 (output layer) and the layer 1 (input layer) relatively receive frequent updates.

layers. Additionally, in Flowers, a considerable disparity in the early stage of training is observed.

In contrast, compared to full finetuning, our BVG-LS method exhibits more consistent bias-variance ratios across layers. Noticeable examples are Pets and DTD, where certain layers (layer 6 and possibly layer 1) experience non-negligible offsets in the bias-variance ratios for the case of full finetuning. These offsets could be the reason for poor full finetuning performance on Pets and DTD datasets, as is evident in Tab. 3. To be more specific, in these cases, the existence of these variance-dominated layers implies that the model becomes to adapt individual training samples. On the other hand, BVG-LS has a mechanism to better maintain consistent bias-variance ratios across layers, as can be observed in Fig. 2.

5 Conclusion

In this paper, we have proposed a fine-tuning strategy called Bias-Variance Guided Layer Selection (BVG-LS), which adaptively selects the layer to be tuned based on the bias-variance ratios of the mini-batch gradients. By considering the bias-variance ratios as an indicator of the risk of overfitting, BVG-LS has selected the layer with the maximum ratio at each training iteration, resulting in the avoidance of overfitting. Experimental results have shown that BVG-LS outperforms full fine-tuning in all seven target tasks for WideResNet-50-2 and in six out of seven target tasks for ViT-S. This highlights the effectiveness of our approach. Finally, we discuss limitations and future work.

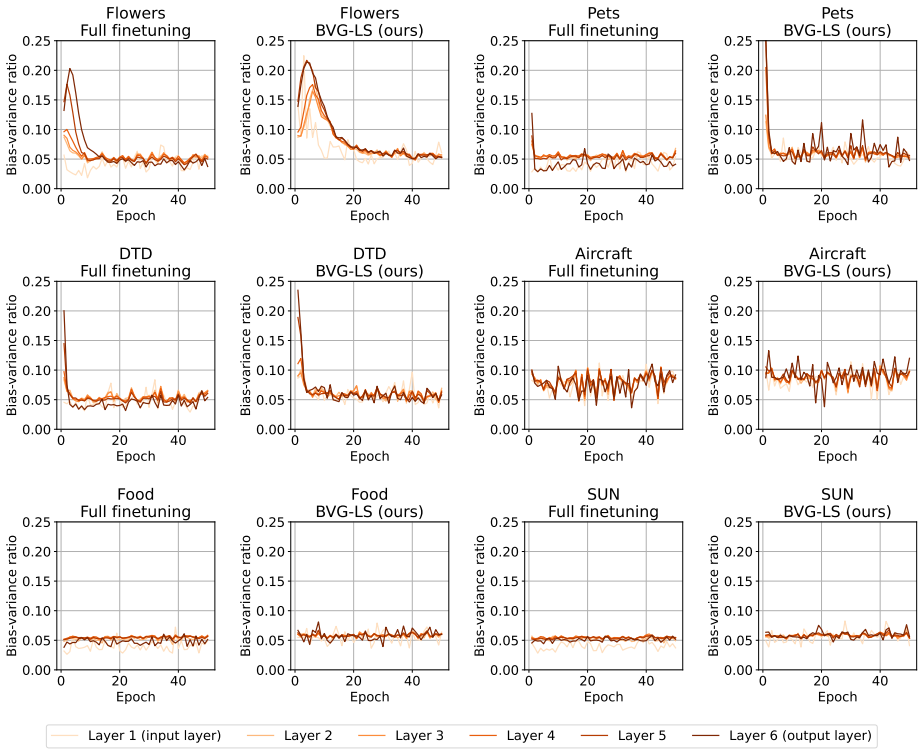


Fig. 6: Layer-wise bias-variance ratios $r^{(l,t)}$ defined by Eq. (2) in full finetuning (1st and 3rd columns) and BVG-LS (2nd and 4th columns). Results of the WideResNet-50-2 models are shown. See the main text for explanation.

Limitation. We focus on layer selection in finetuning without modifying network architectures. While BVG-LS outperformed full finetuning, it does not necessarily always select the best layer for improving performance in target tasks because BVG-LS considers only the first- and second-order statistics. This necessitates addressing higher-order statistics, especially those related to outliers.

Additionally, BVG-LS updated only one layer at each training iteration. While applying a threshold to the bias-variance ratio $r^{(l,t)}$ could allow updating multiple layers simultaneously, this strategy did not improve performance as shown in Table 5. This would be due to gradient conflicts among layers when multiple layers are updated concurrently.

Future work. Our future work includes improving the indicator based on higher-order statistics and analysing loss curvature of each layer. We also plan to exprole a mechanism to skip gradient computations for layers with low bias-variance ratios to reduce computational costs. Furthermore, applying the pro-

Table 5: Performance comparison with the case where a threshold is applied to the bias-variance ratio $r^{(l,t)}$. In the "Threshold" method, instead of updating the layer with the highest $r^{(l,t)}$, all layers with $r^{(l,t)}$ above a certain threshold (set to 0.056 in this case) were updated. The best and second-best results are marked in bold and underlined, respectively.

Dataset	Full finetuning	Linear probing	BVG-LS (ours)	Threshold
Flowers	7.65	12.84	**6.18**	<u>6.28</u>
Pets	15.14	<u>8.22</u>	**6.26**	8.61
DTD	40.37	35.32	**29.20**	<u>31.97</u>
Aircraft	<u>19.34</u>	49.40	**14.60**	19.90
Food	18.01	35.45	**14.19**	<u>16.13</u>
SUN	35.54	36.72	**30.23**	<u>32.44</u>
Average	22.67	29.66	**16.78**	<u>19.22</u>

posed strategy approach to other data types, such as audio and text, would also be a promising next step.

Acknowledgments. This work is an outcome of a research project, Development of Quality Foundation for Machine-Learning Applications, supported by DENSO IT LAB Recognition and Learning Algorithm Collaborative Research Chair (Science Tokyo).

References

1. Agrawal, P., Girshick, R., Malik, J.: Analyzing the performance of multilayer neural networks for object recognition. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VII 13. pp. 329–344. Springer (2014) 3
2. Akiba, T., Shing, M., Tang, Y., Sun, Q., Ha, D.: Evolutionary optimization of model merging recipes. arXiv preprint arXiv:2403.13187 (2024) 4
3. Azizpour, H., Razavian, A.S., Sullivan, J., Maki, A., Carlsson, S.: Factors of transferability for a generic convnet representation. IEEE transactions on pattern analysis and machine intelligence **38**(9), 1790–1802 (2015) 3
4. Bossard, L., Guillaumin, M., Van Gool, L.: Food-101–mining discriminative components with random forests. In: ECCV. pp. 446–461. Springer (2014) 7
5. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531 (2014) 3
6. Chu, B., Madhavan, V., Beijbom, O., Hoffman, J., Darrell, T.: Best practices for fine-tuning visual classifiers to new domains. In: Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14. pp. 435–442. Springer (2016) 3
7. Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., Vedaldi, A.: Describing textures in the wild. In: CVPR. pp. 3606–3613 (2014) 7

8. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. In: International conference on machine learning. pp. 647–655. PMLR (2014) 3
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021) 8
10. Foret, P., Kleiner, A., Mobahi, H., Neyshabur, B.: Sharpness-aware minimization for efficiently improving generalization. In: International Conference on Learning Representations (2021) 2
11. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 580–587 (2014) 3
12. Guo, Y., Shi, H., Kumar, A., Grauman, K., Rosing, T., Feris, R.: Spottune: transfer learning through adaptive fine-tuning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4805–4814 (2019) 2, 4
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR. pp. 770–778. IEEE (2015) 8
14. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012) 2
15. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., Gelly, S.: Parameter-efficient transfer learning for NLP. In: Proceedings of the 36th International Conference on Machine Learning (2019) 4
16. Howard, J., Ruder, S.: Universal language model fine-tuning for text classification. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 328–339 (Jul 2018) 2, 3
17. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: LoRA: Low-rank adaptation of large language models. In: International Conference on Learning Representations (2022) 4
18. Huh, M., Agrawal, P., Efros, A.A.: What makes imagenet good for transfer learning? arXiv preprint arXiv:1608.08614 (2016) 3
19. Iofinova, E., Peste, A., Kurtz, M., Alistarh, D.: How well do sparse imagenet models transfer? In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12266–12276 (2022) 3
20. Jiang, J., Shu, Y., Wang, J., Long, M.: Transferability in deep learning: A survey. CoRR **abs/2201.05867** (2022) 1
21. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: International Conference on Learning Representations (2017) 2
22. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A.A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences **114**(13), 3521–3526 (2017) 2
23. Kolesnikov, A., Beyer, L., Zhai, X., Puigcerver, J., Yung, J., Gelly, S., Houlsby, N.: Big transfer (bit): General visual representation learning. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part V 16. pp. 491–507. Springer (2020) 3

24. Kornblith, S., Shlens, J., Le, Q.V.: Do better imagenet models transfer better? In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2661–2671 (2019) 3
25. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 7
26. Kumar, A., Raghunathan, A., Jones, R.M., Ma, T., Liang, P.: Fine-tuning can distort pretrained features and underperform out-of-distribution. In: International Conference on Learning Representations (2022) 3
27. Lee, Y., Chen, A.S., Tajwar, F., Kumar, A., Yao, H., Liang, P., Finn, C.: Surgical fine-tuning improves adaptation to distribution shifts. In: The Eleventh International Conference on Learning Representations (2023) 4
28. Lester, B., Al-Rfou, R., Constant, N.: The power of scale for parameter-efficient prompt tuning. In: Moens, M.F., Huang, X., Specia, L., Yih, S.W.t. (eds.) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 3045–3059. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (Nov 2021) 4
29. Lin, T.Y., RoyChowdhury, A., Maji, S.: Bilinear cnn models for fine-grained visual recognition. In: 2015 IEEE International Conference on Computer Vision (ICCV). pp. 1449–1457 (2015) 3
30. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11976–11986 (2022) 8
31. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3431–3440 (2015) 3
32. Long, M., Cao, Y., Wang, J., Jordan, M.: Learning transferable features with deep adaptation networks. In: International conference on machine learning. pp. 97–105. PMLR (2015) 3
33. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013) 7
34. Mensink, T., Uijlings, J., Kuznetsova, A., Gygli, M., Ferrari, V.: Factors of influence for transfer learning across diverse appearance domains and task types. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(12), 9298–9314 (2021) 3
35. Nilsback, M.E., Zisserman, A.: Automated flower classification over a large number of classes. In: 2008 Sixth Indian conference on computer vision, graphics & image processing. pp. 722–729. IEEE (2008) 7
36. Nogueira, K., Penatti, O.A., Dos Santos, J.A.: Towards better exploiting convolutional neural networks for remote sensing scene classification. Pattern Recognition **61**, 539–556 (2017) 3
37. Parkhi, O.M., Vedaldi, A., Zisserman, A., Jawahar, C.: Cats and dogs. In: CVPR. pp. 3498–3505. IEEE (2012) 7
38. Romero, M., Interian, Y., Solberg, T., Valdes, G.: Targeted transfer learning to improve performance in small medical physics datasets. Medical physics **47**(12), 6246–6256 (2020) 3
39. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015) 3, 7

40. Salman, H., Ilyas, A., Engstrom, L., Kapoor, A., Madry, A.: Do adversarially robust imagenet models transfer better? Advances in Neural Information Processing Systems **33**, 3533–3545 (2020) 3
41. Sato, I., Ishikawa, K., Liu, G., Tanaka, M.: Breaking inter-layer co-adaptation by classifier anonymization. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 5619–5627. PMLR (09–15 Jun 2019) 2
42. Sato, I., Ryota, Y., Tanaka, M., Inoue, N., Kawakami, R.: Pof: Post-training of feature extractor for improving generalization. In: International Conference on Machine Learning. pp. 19221–19230. PMLR (2022) 2
43. Sharif Razavian, A., Azizpour, H., Sullivan, J., Carlsson, S.: Cnn features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 806–813 (2014) 3
44. Shen, S., Zhou, Y., Wei, B., Chang, E.I.C., Xu, Y.: Tuning stable rank shrinkage: Aiming at the overlooked structural risk in fine-tuning. In: CVPR (2024) 4
45. Shen, Z., Liu, Z., Qin, J., Savvides, M., Cheng, K.T.: Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In: Proceedings of the AAAI conference on artificial intelligence. vol. 35, pp. 9594–9602 (2021) 2, 3
46. Takayama, K., Sato, I., Suzuki, T., Kawakami, R., Uto, K., Shinoda, K.: Smooth transfer learning for source-to-target generalization. In: NeurIPS 2021 Workshop on Distribution Shifts (DistShift) (2021) 1
47. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: Artificial Neural Networks and Machine Learning – ICANN 2018. pp. 270–279 (2018) 1
48. Wortsman, M., Ilharco, G., Kim, J.W., Li, M., Kornblith, S., Roelofs, R., Gontijo-Lopes, R., Hajishirzi, H., Farhadi, A., Namkoong, H., Schmidt, L.: Robust fine-tuning of zero-shot models. In: CVPR (2021) 4
49. Xiao, J., Hays, J., Ehinger, K.A., Oliva, A., Torralba, A.: Sun database: Large-scale scene recognition from abbey to zoo. In: CVPR. pp. 3485–3492. IEEE (2010) 7
50. Xu, R., Luo, F., Zhang, Z., Tan, C., Chang, B., Huang, S., Huang, F.: Raise a child in large language model: Towards effective and generalizable fine-tuning. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 9514–9528 (Nov 2021) 4
51. Yamada, Y., Otani, M.: Does robustness on imagenet transfer to downstream tasks? In: CVPR (2022) 2
52. Yamaguchi, S., Kanai, S., Adachi, K., Chijiwa, D.: Adaptive random feature regularization on fine-tuning deep neural networks. In: CVPR (2024) 4
53. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? Advances in neural information processing systems **27** (2014) 2, 3
54. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016) 8
55. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13. pp. 818–833. Springer (2014) 2
56. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning requires rethinking generalization. In: ICLR (2017) 1
57. Zhang, Z., Zhang, Q., Gao, Z., Zhang, R., Shutova, E., Zhou, S., Zhang, S.: Gradient-based parameter selection for efficient fine-tuning. In: CVPR (2024) 4