Domain-Aware Fine-Tuning: Enhancing Neural Network Adaptability

Seokhyeon Ha[1], **Sunbeom Jeong**[1], **Jungwoo Lee**[1,2]
[1] Seoul National University
[2] HodooAI Lab
{aoxm1231, sb3991, junglee}@snu.ac.kr

Abstract

Fine-tuning pre-trained neural network models has become a widely adopted approach across various domains. However, it can lead to the distortion of pre-trained feature extractors that already possess strong generalization capabilities. Mitigating feature distortion during adaptation to new target domains is crucial. Recent studies have shown promising results in handling feature distortion by aligning the head layer on in-distribution datasets before performing fine-tuning. Nonetheless, a significant limitation arises from the treatment of batch normalization layers during fine-tuning, leading to suboptimal performance. In this paper, we propose Domain-Aware Fine-Tuning (DAFT), a novel approach that incorporates batch normalization conversion and the integration of linear probing and fine-tuning. Our batch normalization conversion method effectively mitigates feature distortion by reducing modifications to the neural network during fine-tuning. Additionally, we introduce the integration of linear probing and fine-tuning to optimize the head layer with gradual adaptation of the feature extractor. By leveraging batch normalization layers and integrating linear probing and fine-tuning, our DAFT significantly mitigates feature distortion and achieves improved model performance on both in-distribution and out-of-distribution datasets. Extensive experiments demonstrate that our method outperforms other baseline methods, demonstrating its effectiveness in not only improving performance but also mitigating feature distortion.

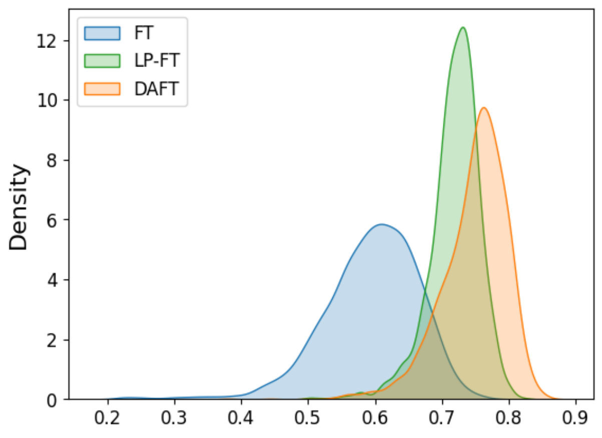
Introduction

Transferable neural network models have seen significant advancements in various domains, including computer vision (He et al. 2020; Chen et al. 2020a; Radford et al. 2021), natural language processing (Kenton and Toutanova 2019; Yang et al. 2019), and speech recognition (Baevski et al. 2020; Gulati et al. 2020). These models, trained on large-scale datasets with substantial computational resources, demonstrate exceptional performance and generalization capabilities, making them widely used in transfer learning. By employing these pre-trained models, transfer learning can substantially reduce the time and resources required for training and data collection, while also enhancing performance compared to training from scratch.

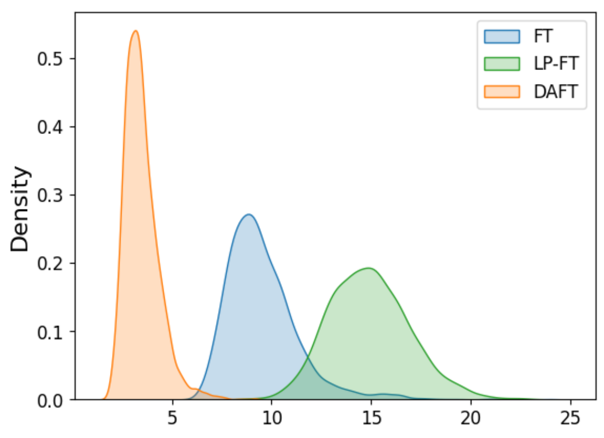
Copyright © 2024, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

When transferring knowledge from a pre-trained model, the feature extractor is first initialized with the pre-trained model. Subsequently, there are two primary optimization methods commonly employed: linear probing (LP) and fine-tuning (FT). In LP, only the head layer is optimized, while the feature extractor is fixed during training. On the other hand, FT involves optimizing both the feature extractor and the linear head layer simultaneously. While FT generally outperforms LP on in-distribution (ID) datasets, recent findings show that FT performs worse on out-of-distribution (OOD) datasets compared to LP. It is due to feature distortion in pre-trained features that possess good generalization capabilities. To mitigate feature distortion, LP-FT was proposed as a two-stage training approach (Kumar et al. 2022). In the first stage, LP aligns the head layer with the ID subspace, and in the second stage, FT is performed with a small learning rate to mitigate feature distortion. This allows LP-FT to achieve improved performance on both ID and OOD datasets.

However, LP-FT still has several limitations. One of the key issues is the treatment of batch normalization layers during the FT stage in the LP-FT framework. During the FT stage, all batch normalization layers are switched to test mode, where fixed statistics are employed instead of batch statistics. While this prevents the alteration of batch normalization statistics during training, potentially mitigating feature distortion, it also harms beneficial effects such as improved gradient flow and reduced internal covariate shift (Ioffe and Szegedy 2015). Consequently, the overall optimization process is degraded, leading to suboptimal model performance. Furthermore, the initial LP stage in LP-FT optimizes the head layer with a feature extractor that has not been adapted to the target domain, which might not be ideal for domain adaptation. Moreover, LP-FT utilizes batch normalization with fixed test statistics instead of batch statistics, which can lead to gradient exploding if not accompanied by a significantly small learning rate. Therefore, LP-FT employs a smaller learning rate during the FT stage compared to standard FT. It limits the optimization of the head layer during the FT stage, especially when there is a significant dissimilarity in data distribution between the source and target domains. These limitations call for a more effective approach that can adequately address feature distortion while effectively performing batch normalization.



(a) Cosine Similarity



(b) L2 Distance

Figure 1: Distribution of similarity measures for three different fine-tuning methods. We fine-tune the pre-trained ResNet-50 model from MoCo-v2 (Chen et al. 2020b) on fMoW (Christie et al. 2018) dataset. First, we extract features from the test data before applying each fine-tuning method. Then, after completing the fine-tuning process, we compute the similarity measures between the pre-fine-tuning and post-fine-tuning features for each method. The similarity measures, including cosine similarity and L2 distance, are computed on the fMoW test dataset. Notably, our DAFT exhibits the least distortion in the pre-trained features across both similarity measures.

To address these challenges, we present a novel method named Domain-Aware Fine-Tuning (DAFT). Our method is built upon two core techniques: batch normalization conversion and the integration of LP and FT into a single stage. The batch normalization conversion method modifies batch normalization layers to better suit the target domain prior to fine-tuning. It makes batch normalization more effective, reducing feature distortion by adjusting the statistics and parameters to the target domain. In addition, our DAFT integrates LP and FT into a single stage, where the linear head is optimized with gradually adapting features from the target domain. This strategy mitigates the issue of the head layer being optimized solely using a feature extractor that has not yet adapted to the target domain, resulting in improved optimization.

Through these techniques, our DAFT effectively reduces feature distortion and achieves enhanced performance compared to other baseline methods. The effectiveness of our method is demonstrated through similarity measures computed between features before and after adaptation. As shown in Figure 1, our DAFT consistently exhibits higher feature similarity compared to existing methods such as FT and LP-FT. Furthermore, we conduct extensive experiments to compare our proposed method against other baseline methods. These results demonstrate the superior performance of the proposed DAFT with higher accuracy and less feature distortion.

In summary, our main contributions are as follows.

- We propose a novel batch normalization conversion technique, which improves batch normalization during fine-tuning with reducing modifications to the neural network.
- The integration of LP and FT is proposed, allowing the head layer to be optimized with the gradual adaptation of the feature extractor.
- Through extensive experiments, we demonstrate the superior performance of our proposed method compared to other baseline approaches.

Related Works

Head Initialization for Fine-Tuning LP-FT, a two-stage transfer learning method, initializes the head layer with parameters obtained from LP prior to executing FT (Kumar et al. 2022). This highlights that random initialization of the head layer has the potential to cause feature distortion in FT, leading to degraded generalization performance. Nonetheless, the performance of LP-FT remains limited even with aligned head initialization, calling for further research on better head initialization methods. One promising technique is to stop the LP stage early before it reaches convergence, resulting in a non-converged head that improves performance during the subsequent FT stage (Ren et al. 2023). Additionally, applying hardness-promoting augmentation during the LP stage can help mitigate feature distortion and also simplicity bias, leading to enhanced generalization performance (Trivedi, Koutra, and Thiagarajan 2023). Although these head initialization techniques have shown effectiveness in boosting the performance of the FT stage, they require a separate training stage to initialize the head layer before the FT stage. In contrast, our proposed DAFT integrates head adaptation seamlessly without the need for a distinct initialization stage.

Other Modifications for Fine-Tuning There are several other techniques to improve the performance of FT. FLYP utilizes contrastive learning with additional text data (Goyal et al. 2023). Side-tuning involves fixing the feature extractor to preserve pre-trained features and combining it with a small side network, then training only the side network and the head layer (Zhang et al. 2020). Other methods combine FT with the regularization of pre-trained parameters (Xuhong, Grandvalet, and Davoine 2018; Li et al. 2019; Gouk, Hospedales, and Pontil 2021; Li and Zhang 2021), or the use of different learning rates for each layer (Howard and Ruder 2018; Yang et al. 2019; Clark et al. 2020). Our method falls under the category of employing different learning rates, as we use a larger learning rate for the head layer. However, unlike general methods that train the head layer with a learning rate approximately 10 times larger than that of the feature extractor, our method uses separate learning rates for the head layer and the feature extractor. This allows the head layer to align better with the ID dataset during the early stage of training and to converge with the adapting feature extractor.

Batch Normalization in Transfer Learning Batch normalization is a well-known technique for covariance shift reduction and better stable training (Ioffe and Szegedy 2015). However, in transfer learning scenarios, where the data distribution between the source domain and target domain may significantly differ, the movement of statistics within batch normalization layers can result in severe feature distortion. As a solution, many methods choose to freeze the batch normalization statistics during fine-tuning to alleviate feature distortion (Kumar et al. 2022; Ren et al. 2023; Trivedi, Koutra, and Thiagarajan 2023). On the other hand, some methods leverage batch normalization with domain-specific statistics to encourage the learning of more generalized features (Wang et al. 2019; Chang et al. 2019) or even use statistics from test batches to enhance robustness against domain shift (Mirza et al. 2022; Lim et al. 2023). In contrast to existing approaches, our method aims to mitigate distortion of pre-trained features while still benefiting from batch normalization during training.

Method

In this section, we introduce our approach, Domain-Aware Fine-Tuning (DAFT), which incorporates converting batch normalization and integrating LP and FT.

Converting Batch Normalization

Batch Normalization (BN) The Batch Normalization (BN) technique (Ioffe and Szegedy 2015) was introduced based on the observation that network training tends to converge faster when its inputs are whitened (LeCun et al. 2002; Wiesler and Ney 2011). BN operates in two modes: training mode and test mode. In the training mode, it first computes batch statistics, μ and σ^2, for the input features z. Then, scaling and shifting parameters, γ and β, are applied to scale and shift the normalized values. The normalization process is represented as follows:

$$\hat{y} = \gamma \cdot \frac{z - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta, \tag{1}$$

where ϵ is a small constant for numerical stability. During training, BN computes the mean μ and variance σ^2 over a mini-batch of training data as

$$\mu = \frac{1}{m}\sum_{i=1}^{m} z_i, \quad \sigma^2 = \frac{1}{m}\sum_{i=1}^{m}(z_i - \mu)^2, \tag{2}$$

where m is the batch size. In the test mode, BN uses test statistics estimates, M and Σ, instead of μ and σ. Test statistics are obtained by using the moving averages of the train batch statistics. These moving averages are continuously updated during training to ensure consistent normalization during the test mode without the need for additional statistics computation.

BN Transfer Issue Due to the presence of *dataset bias* or *domain shift* (Quinonero-Candela et al. 2008), significant differences arise in the input distribution of batch normalization between the source and the target domains. Let $\mathrm{M}s$ and Σs respectively represent the test mean and test variance of batch normalization, pre-trained on the source domain. When training a pre-trained model on a new target domain, the values of M_s and Σ_s undergo considerable updates, leading to a significant deviation from their original values. Moreover, the learning parameters also experience substantial adjustments during training to adapt and converge with the target domain statistics. These significant changes in batch normalization are one of the main factors that cause feature distortion during fine-tuning.

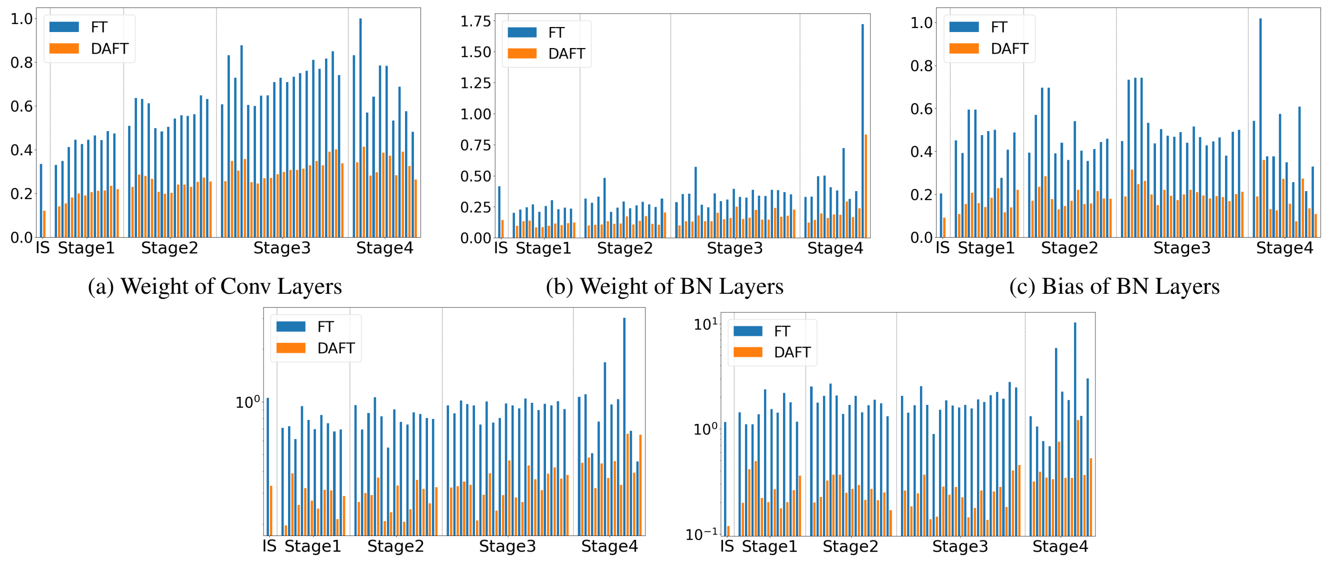
Batch Normalization Conversion While many recent studies have addressed the BN transfer issue by introducing additional statistics and parameters for the new domain (Chang et al. 2019; Lim et al. 2023), we handle this issue by simply converting the existing statistics and parameters without introducing additional ones. We introduce our BN conversion to effectively prevent the distortion of pre-trained feature extractors. Before starting the training process, we first compute batch statistics, μ_t and σ_t, for each mini-batch of the target training data. Next, we compute the new unbiased statistics, M_t and Σ_t, as follows:

$$\mathrm{M}_t = \mathrm{E}_{\mathcal{B}_t}[\mu_t], \quad \Sigma_t = \frac{m}{m-1}\mathrm{E}_{\mathcal{B}_t}[\sigma_t^2], \tag{3}$$

where $\mathcal{B}_t$ represents mini-batches of target training data. With these statistics estimated from the target domain, we can reformulate the batch normalization of the pre-trained model as follows:

$$\begin{aligned}
\hat{y} &= \gamma_s \cdot \frac{z - \mathrm{M}_s}{\sqrt{\Sigma_s + \epsilon}} + \beta_s \\
&= \gamma_s \cdot \frac{\sqrt{\Sigma_t + \epsilon}}{\sqrt{\Sigma_s + \epsilon}} \cdot \frac{z - \mathrm{M}_t + \mathrm{M}_t - \mathrm{M}_s}{\sqrt{\Sigma_t + \epsilon}} + \beta_s, \\
&= (\gamma_s \frac{\sqrt{\Sigma_t + \epsilon}}{\sqrt{\Sigma_s + \epsilon}}) \cdot \frac{z - \mathrm{M}_t}{\sqrt{\Sigma_t + \epsilon}} + (\beta_s + \gamma_s \frac{\mathrm{M}_t - \mathrm{M}_s}{\sqrt{\Sigma_s + \epsilon}}) \\
&= \gamma_t \cdot \frac{z - \mathrm{M}_t}{\sqrt{\Sigma_t + \epsilon}} + \beta_t,
\end{aligned} \tag{4}$$

where γ_s and β_s are the learning parameters of batch normalization that are pre-trained on the source domain. According to this reformulation, we define the new batch normalization parameters, γ_t and β_t, with the target domain



(a) Weight of Conv Layers (b) Weight of BN Layers (c) Bias of BN Layers

(d) Mean of BN Layers (e) Variance of BN Layers

Figure 2: Comparison of relative changes in learning parameters (2a, 2b, 2c) and BN statistics (2d, 2e) between FT and our DAFT on the fMoW dataset. We utilize the pre-trained ResNet-50 model from MoCo-v2 as the feature extractor and conduct each fine-tuning method. The ResNet-50 architecture consists of an Input Stem and 4 subsequent stages, with each stage indicated on the x-axis from left to right. 'IS' on the x-axis represents the Input Stem, and all layers within each stage are further indicated on the x-axis sequentially from left to right. The relative changes are computed between the initial values before each fine-tuning and the final values after each fine-tuning process. Note that the relative changes of BN statistics are represented in log scale.

statistics as follows:

$$\gamma_t = \gamma_s \cdot \frac{\sqrt{\Sigma_t + \epsilon}}{\sqrt{\Sigma_s + \epsilon}}, \quad \beta_t = \beta_s + \gamma_s \cdot \frac{\mathrm{M}_t - \mathrm{M}_s}{\sqrt{\Sigma_s + \epsilon}}. \tag{5}$$

After computing these new parameters, we replace γ_s and β_s in the pre-trained model with γ_t and β_t, respectively. Additionally, the statistics M_s and Σ_s are replaced by M_t and Σ_t. After converting all batch normalization layers in the pre-trained model, we start training on the new target domain. With the converted BN parameters, the pre-trained model requires only slight adjustments to the batch statistics of the target domain. Furthermore, the movement of M_t and Σ_t can be largely reduced since the batch statistics of the target training data are already close to M_t and Σ_t. As a result, the converted BN substantially mitigates feature distortion, leading to improved training performance on the target domain. The overall process of our BN conversion is summarized in Algorithm 1.

Utilizing our BN conversion technique, neural networks undergo more stable adjustments that aid in mitigating the distortion of pre-trained features, as shown in Figure 1. For a comprehensive understanding of how neural network evolves during adaptation, we compute the norm of the relative change using the equation:

$$\frac{\|\Delta W\|_2}{\|W\|_2} = \frac{\|\widetilde{W} - W\|_2}{\|W\|_2}, \tag{6}$$

where $\widetilde{W}$ represents the parameter values after the training is completed, and W represents the initial parameter values before training begins. To conduct our evaluation, we employ a pre-trained ResNet-50 model from MoCo-v2 (Chen et al. 2020b) and perform two types of training on fMoW (Christie et al. 2018) dataset: fine-tuning without our BN conversion (FT) and fine-tuning with our BN conversion (DAFT). For each case, we compute the norm of the relative change for all layers of the ResNet-50 model. It is important to note that W for FT refers to the value of the pre-trained model, whereas W for DAFT refers to the value after applying BN conversion to the pre-trained model. As illustrated in Figure 2, we can clearly observe that DAFT leads to significantly smaller relative changes compared to FT. This reduction is evident not only in the parameters of all layers but also in the statistics of all BN layers. This compelling evidence substantiates the effectiveness of our BN conversion technique, enabling the model to adapt more effectively to the target domain while inducing minor alterations.

Integrating LP and FT

Our BN conversion technique can be applied to various transfer learning algorithms for neural networks that incorporate BN layers. However, considering the limitations of the recent LP-FT method, we integrate LP and FT in a single stage to leverage BN conversion more effectively.

Limitation of LP-FT Let θ represent the parameters of the feature extractor denoted as f_θ, initialized with a pre-trained model. The classifier with parameters w is denoted as

Algorithm 1: Batch Normalization Conversion

```
Require: Pre-trained model with BN parameters $\gamma_s$, $\beta_s$, and statistics $\mathrm{M}_s$, $\Sigma_s$ from the source domain, Target training data with mini-batches $\mathcal{B}_t$, each containing $m$ elements;
    Set pre-trained model to test mode
    for each mini-batch $\mathcal{B} \in \mathcal{B}_t$ do
        Compute batch statistics $\mu_t$ and $\sigma_t$ with $\mathcal{B}$ for all BN layers
    end for
    Compute the unbiased estimates of statistics for all BN layers: $M_t = \mathrm{E}_{\mathcal{B}_t}[\mu_t]$, $\Sigma_t = \frac{m}{m-1}\mathrm{E}_{\mathcal{B}_t}[\sigma_t^2]$
    Compute the new parameters for all BN layers: $\gamma_t = \gamma_s \cdot \frac{\sqrt{\Sigma_t+\epsilon}}{\sqrt{\Sigma_s+\epsilon}}$, $\beta_t = \beta_s + \gamma_s \cdot \frac{M_t - M_s}{\sqrt{\Sigma_s+\epsilon}}$
    Replace parameters $\gamma_s$ with $\gamma_t$, and $\beta_s$ with $\beta_t$ in the pre-trained model for all BN layers
    Replace the statistics $M_s$ with $M_t$, and $\Sigma_s$ with $\Sigma_t$ in the pre-trained model for all BN layers
    Start fine-tuning on the target domain
```

g_w. While it is commonly a linear layer, it could also consist of multiple layers. In the LP-FT approach, the head classifier g_w is initialized using the value optimized by LP, and then FT is performed with a very small learning rate η. As a result, the head layer undergoes only minor changes during FT and remains nearly unchanged after the FT stage. This implies that the head layer optimization primarily occurs during the LP stage, and it is not optimized in conjunction with the feature extractor's adaptation to the target domain. This limitation hinders performance, and it becomes particularly critical when there is a significant difference in data distribution between the source and target domains. To address this issue, we introduce integrating LP and FT into a single stage, allowing the head layer to be optimized with gradually adapting features.

Integrated LP-FT In order to integrate the LP and FT stages, we introduce separate learning rates, η_θ and η_w, for the feature extractor f_θ and the head layer g_w. While the conventional FT also suggests using a different learning rate for the head layer that is 10 times larger than that of the feature extractor, we independently determine the optimized learning rates for each component. To efficiently determine these optimized learning rates, we adopt a two-step procedure, sweeping over each learning rate. In the first step, we sweep over the learning rates η_θ for the feature extractor with fixing η_w. Having determined the optimized learning rate η_θ for the feature extractor, we move on to the second step, where we sweep over the learning rates η_w while utilizing the previously determined optimized learning rate η_θ. By following this process, we integrate the FT and LP stages into a single approach, which distinguishes our approach from LP-FT. Moreover, this strategy significantly reduces the number of hyperparameter combinations and streamlines the search for proper learning rates.

Additionally, we employ a zero initialization strategy for the head layer. This approach is aimed at reducing the influence of gradients on the feature extractor during the initial training phases. By initializing the head layer with zeros, we try to avoid making significant changes to the feature extractor until the head layer has started to converge to a reasonable solution. However, in cases where the head layer comprises multiple layers, we opt for random initialization instead of zero values to enable gradient computation. Detailed distinctions between our DAFT and other fine-tuning techniques, including FT and LP-FT, are summarized in Table 1.

Method	Head Init	LR for FT	BN mode	BN Conversion
FT	Random	$\eta_\theta = \eta_w$	train	X
LP-FT	LP	$\eta_\theta = \eta_w$	test	X
DAFT	Zero	$\eta_\theta \neq \eta_w$	train	O

Table 1: Summary of details for our DAFT and other fine-tuning methods.

Experiments

In this section, we present the results of various experiments conducted to evaluate the effectiveness of our proposed method. We first present the experimental results on classification tasks and segmentation tasks. Additionally, we provide the results of additional experiments, including robustness evaluations and ablation studies.

Experiments on Classification Task

Dataset We conduct experiments on various classification tasks and evaluate both In-Distribution (ID) and Out-of-Distribution (OOD) accuracy using the following datasets.

- **CIFAR-10** (Krizhevsky et al. 2009): A dataset that contains 10 categories of objects. For the OOD dataset, we use two additional datasets, **CIFAR-10.1** (Recht et al. 2018) and **STL** (Coates, Ng, and Lee 2011). CIFAR-10.1 is a subset of the TinyImage dataset (Birhane and Prabhu 2021) and has the same categories as CIFAR-10. STL has the same categories as CIFAR-10 except for 'monkey' instead of 'frog'. Therefore, we remove the 'monkey' class in STL to align with CIFAR-10 categories. For data augmentation, we resize the input image to 224x224 and apply random horizontal flip.
- **Entity-30** and **Living-17** (Santurkar, Tsipras, and Madry 2020): Part of the BREEDS benchmarks, which are subpopulation shift datasets constructed using ImageNet. Entity-30 contains 30 categories of objects, and Living-17 contains 17 categories of animals. Each dataset consists of source and target domains. For both Entity-30 and Living-17, we use the source domain as the ID dataset and the target domain as the OOD dataset. Data augmentation involves RandomResizedCrop to 224x224 and random horizontal flip.
- **DomainNet** (Peng et al. 2019): A dataset that includes 345 categories of common objects in six different domain including Clipart, Infograph, Painting, Quickdraw,

Model	Method	CIFAR-10			Entity-30		Living-17		DomainNet		fMoW	
		ID	CIFAR-10.1	STL	ID	OOD	ID	OOD	ID	OOD	ID	OOD
MoCo	LP	91.78	82.53	86.01	91.35	64.10	96.27	82.06	73.67	64.41	49.12	34.92
	FT	97.66	92.95	81.18	93.39	62.48	96.98	78.41	87.77	62.19	60.72	41.04
	LP-FT	97.66	93.60	90.79	93.80	62.03	97.55	82.53	85.56	70.89	55.24	38.50
	DAFT	**97.84**	**94.18**	**92.36**	**94.54**	**65.11**	**97.84**	**84.35**	**89.31**	**74.67**	**61.91**	**43.29**
CLIP	LP	87.87	77.62	84.82	91.11	63.99	94.61	**79.57**	89.13	82.29	49.85	31.24
	FT	93.88	87.07	69.50	93.74	60.28	94.69	66.96	84.67	53.91	54.83	31.65
	LP-FT	94.02	87.77	87.01	93.29	62.62	95.73	77.96	91.27	82.53	60.73	39.93
	DAFT	**95.49**	**89.52**	**87.51**	**94.40**	**64.34**	**96.47**	78.24	**91.72**	**82.57**	**64.65**	**43.40**
SWAV	LP	93.24	85.23	89.58	92.05	63.91	96.67	80.06	75.13	59.71	47.29	29.74
	FT	97.55	93.25	88.26	94.96	63.03	97.61	80.65	88.34	66.69	66.06	45.67
	LP-FT	97.50	92.60	**91.44**	94.67	61.78	97.47	80.84	85.19	67.38	58.27	37.62
	DAFT	**97.77**	**93.82**	89.96	**95.07**	**63.96**	**97.88**	**84.06**	**88.80**	**69.03**	**66.93**	**46.14**

Table 2: ID and OOD accuracies (%) of three different pretrained models on five different dataset. The upper part of the first row represents the training data and the lower part represents the test data. CIFAR-10.1 and STL are used as OOD datasets for CIFAR-10, as there is no dedicated OOD dataset for CIFAR-10. Best results are highlighted in bold.

Real, and Sketch. However, due to labeling noise in DomainNet, we used a subset of DomainNet (Tan, Peng, and Saenko 2020) containing 40 categories of common objects from the Sketch, Real, Clipart, and Painting domains. Sketch domain is used for the ID dataset and the rest of the domains (Real, Clipart, Painting) are used for the OOD dataset. Data augmentation includes resizing the images to 256x256, random cropping to 224x224, and applying random horizontal flip.

- **fMoW** (Christie et al. 2018): A remote sensing dataset that contains 62 categories of satellite images in five different regions including Asia, Europe, Africa, Americas, and Oceania. For our evaluation, we use images from the Americas region as the ID dataset, and images from the Europe and Africa regions are used for the OOD dataset. As the fMoW data size is 224x224, we apply only random horizontal flip for data augmentation.

Pretrained Models To ensure the generalizability of our method across different pre-trained models, we utilize three different models based on the ResNet-50 architecture: MoCo-v2 (Chen et al. 2020b), CLIP (Radford et al. 2021), and SWAV (Caron et al. 2020). Exceptionally, for the fMoW dataset, we use MoCo-TP (Ayush et al. 2021) instead of MoCo-v2 as the pre-trained model.

Training Details Consistent with LP-FT (Kumar et al. 2022), we use ℓ_2-regularized logistic regression classifier for linear probing and also choose the best ℓ_2-regularization hyperparameter based on ID validation accuracy. For fine-tuning, we employ an SGD classifier with a cosine learning rate schedule and a batch size of 64. We fine-tune for 20 epochs on CIFAR-10, Entity-30, and Living-17, but extend the fine-tuning to 50 epochs on DomainNet and fMoW due to their limited number of images. Early stopping is applied based on the ID validation accuracy.

Method	VOC	Cityscapes
LP	75.85	65.69
FT	78.47	74.42
LP-FT	77.13	66.41
DAFT	**79.23**	**76.03**

Table 3: Mean Intersection over Union (%) for segmentation datasets. Best results are highlighted in bold.

Results Our experimental results on five distinct datasets are summarized in Table 2. All of results are averaged over three separate runs. First, we observe that FT tends to exhibit superior performance in ID accuracy compared to LP, while LP tends to outperform FT in OOD accuracy. Across three diverse pre-trained models, our DAFT consistently achieves higher ID accuracy on all datasets in comparison to other baseline methods. Additionally, DAFT consistently exhibits higher OOD accuracy across most of the datasets.

Experiments on Segmentation Task

Dataset For the segmentation task, we use the following datasets:

- **Pascal VOC2012** (Everingham et al. 2015): This dataset consists of 21 classes for semantic segmentation, including 20 object classes (e.g., person, car, bicycle, etc.) and an additional background class. To augment the data, we apply random scaling between 0.5 and 2.0, random crop of size 513x513, and random horizontal flip. Additionally, we use the augmented dataset with extra annotations (Abadi et al. 2016).
- **Cityscapes** (Cordts et al. 2016): This dataset includes 19 classes for semantic segmentation, comprising 18 object classes (e.g., cars, pedestrians, buildings, roads, etc.) and an additional background class. To augment the data, we

Model	Method	Noise			Blur				Weather				Digital				mCE
		Gauss.	Shot	Impulse	Defocus	Glass	Motion	Zoom	Snow	Frost	Fog	Bright	Cont.	Elastic	Pixel	JPEG	
MoCo	LP	47.5	40.1	54.2	19.5	50.8	33.4	23.5	22.8	23.6	23.7	10.1	18.3	28.7	30.5	25.7	30.2
	FT	56.1	41.9	**45.5**	9.9	41.0	17.1	12.4	9.3	11.9	7.4	3.2	9.8	**12.6**	27.2	20.9	21.7
	LP-FT	46.5	36.0	48.3	8.1	**38.2**	**14.6**	**10.4**	**9.1**	11.3	**7.1**	3.3	11.0	12.9	**24.6**	**19.2**	20.0
	DAFT	**37.4**	**28.8**	46.7	**8.0**	39.8	16.3	10.8	9.3	**10.9**	8.2	**3.2**	**9.1**	14.4	28.5	19.5	**19.4**
CLIP	LP	65.1	57.7	56.2	24.5	57.4	35.3	27.8	26.5	27.9	23.4	15.4	31.8	30.1	28.5	33.8	36.1
	FT	**46.5**	**35.1**	**37.2**	17.6	52.2	24.8	22.0	19.6	22.2	12.7	8.4	24.7	19.0	**27.6**	**18.7**	25.9
	LP-FT	57.3	46.1	48.5	14.4	51.6	24.4	18.3	15.8	18.2	11.6	8.1	23.8	20.4	33.9	27.3	28.0
	DAFT	61.6	48.1	40.7	**13.6**	**47.0**	**22.0**	**16.9**	**13.8**	**15.9**	**9.7**	**6.2**	**18.4**	**17.6**	32.0	23.0	**25.8**
SWAV	LP	50.0	42.8	60.2	17.4	45.8	27.2	20.2	19.3	22.3	19.6	8.6	12.5	25.0	27.8	25.2	28.3
	FT	51.7	38.6	46.9	10.1	39.1	16.8	11.5	**8.7**	10.6	7.7	3.6	10.9	**12.4**	29.2	21.0	21.3
	LP-FT	50.1	37.7	**36.6**	**8.7**	**34.0**	**15.4**	**10.4**	8.8	11.1	**6.6**	3.4	9.4	12.5	**25.3**	20.5	19.4
	DAFT	**41.4**	**30.2**	46.9	8.9	36.1	16.0	10.9	8.9	**10.0**	7.6	**3.1**	**7.7**	12.8	25.4	**19.5**	**19.0**

Table 4: Corruption Error (%) for 15 types of corruptions. The mean Corruption Error (mCE) is calculated for all corruptions. A lower value of Corruption Error indicates better performance. Best results are highlighted in bold.

use random crop of size 768x768, color jitter with brightness 0.5, contrast 0.5, saturation 0.5, and hue 0, and random horizontal flip.

Training Details For the segmentation task, we adopt the DeepLabv3+ (Chen et al. 2018) framework and initialize its backbone with a pre-trained ResNet-50 model from MoCo-v2. During fine-tuning, we use the SGD optimizer with a batch size of 16 and a polynomial learning rate schedule with power 0.9. We conduct the fine-tuning process for a total of 30,000 iterations, and the final model is evaluated thereafter.

Results Table 3 presents the results of our experiments on the segmentation task. Our DAFT demonstrates improved performance compared to other baseline methods for both the VOC and Cityscapes datasets. LP-FT also shows some enhancement by employing an LP-trained model, but it still achieves inferior performance compared to FT. These results show the significance of BN in the context of the segmentation task as well.

Additional Experiments

Robustness In addition to ID and OOD test, we also evaluate the robustness of our method. After fine-tuning on the CIFAR-10 dataset, we evaluate the model's performance on the CIFAR-10-C dataset (Hendrycks and Dietterich 2019). The CIFAR-10-C dataset comprises 15 types of corruptions grouped into four main categories: noise, blur, weather, and digital. Each corruption type is present at five different levels of severity. We calculate the Corruption Error for all levels of corruptions and then compute their average. The results of the robustness assessment are presented in Table 4. Interestingly, LP outperforms FT in terms of OOD accuracy, but it shows weaker performance in terms of robustness. And, our DAFT achieves lower average errors across various corruptions compared to other methods. Since this robustness experiment is conducted for the methods whose results are presented in Table 2, the Corruption Error values are also obtained as an average over three runs.

Model	Method	CIFAR-10		
		ID	CIFAR-10.1	STL
MoCo	FT	97.66	92.95	81.18
	+ BN conversion	97.59	93.27	84.05
	LP-FT	97.66	93.60	90.79
	+ BN conversion	97.31	93.00	91.66
	Integrated LP-FT	97.63	94.12	90.52
	+ BN conversion	97.84	94.18	92.36
CLIP	FT	93.88	87.07	69.50
	+ BN conversion	95.40	88.8	81.47
	LP-FT	94.02	87.77	87.01
	+ BN conversion	94.70	88.50	85.49
	Integrated LP-FT	94.29	87.32	73.12
	+ BN conversion	95.49	89.52	87.51
SWAV	FT	97.55	93.25	88.26
	+ BN conversion	97.62	93.62	89.00
	LP-FT	97.50	92.60	91.44
	+ BN conversion	97.61	93.45	89.35
	Integrated LP-FT	97.28	92.53	89.46
	+ BN conversion	97.77	93.82	89.96

Table 5: BN conversion effect for each fine-tuning method.

Ablation Study Table 5 presents the results of our ablation study on BN conversion and integrated LP-FT. We conducted the study using a pre-trained ResNet-50 model from MoCo-v2 (Chen et al. 2020b) on the CIFAR-10 dataset. To evaluate the effectiveness of BN conversion alone, we applied it to each fine-tuning method, including FT, LP-FT, and integrated LP-FT. The results demonstrate that BN conversion is effective for most of OOD tests with all fine-tuning methods, indicating its importance in improving performance on OOD datasets. Additionally, we observe that integrated LP-FT without BN conversion performs similarly to LP-FT, revealing the limitations of the two-stage optimization and emphasizing the critical role of BN in the overall optimization process.

Conclusion

In this paper, we introduce Domain-Aware Fine-Tuning (DAFT), a novel approach designed to enhance the adaptability and performance of fine-tuned neural networks. Our method optimizes performance and minimizes network modification by aligning batch normalization layers with the target domain. Additionally, the integration of LP and FT allows the head layer to be optimized with gradually adapting features. The widespread use of batch normalization layers in many practical networks makes DAFT a valuable solution for real-world applications. Overall, DAFT bridges the gap between pre-trained models and new target domains, which contributes to improved model performance and generalization.

Acknowledgments

This work was supported in part by National Research Foundation of Korea (NRF, 2021R1A2C2014504(20%) and 2021M3F3A2A02037893(20%)), in part by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT) (2021-0-00106(15%), 2021-0-02068(15%), 2021-0-01080(15%), and 2021-0-01341(15%)), and in part by AI Graduate School Program, CAU, INMAC and BK21 FOUR program.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467.*

Ayush, K.; Uzkent, B.; Meng, C.; Tanmay, K.; Burke, M.; Lobell, D.; and Ermon, S. 2021. Geography-aware self-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 10181–10190.

Baevski, A.; Zhou, Y.; Mohamed, A.; and Auli, M. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, 33: 12449–12460.

Birhane, A.; and Prabhu, V. U. 2021. Large image datasets: A pyrrhic win for computer vision? In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1536–1546. IEEE.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33: 9912–9924.

Chang, W.-G.; You, T.; Seo, S.; Kwak, S.; and Han, B. 2019. Domain-specific batch normalization for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 7354–7362.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, 801–818.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, X.; Fan, H.; Girshick, R.; and He, K. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297.*

Christie, G.; Fendley, N.; Wilson, J.; and Mukherjee, R. 2018. Functional map of the world. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6172–6180.

Clark, K.; Luong, M.-T.; Le, Q. V.; and Manning, C. D. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555.*

Coates, A.; Ng, A.; and Lee, H. 2011. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 215–223. JMLR Workshop and Conference Proceedings.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Everingham, M.; Eslami, S. A.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2015. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111: 98–136.

Gouk, H.; Hospedales, T. M.; and Pontil, M. 2021. Distance-Based Regularisation of Deep Networks for Fine-Tuning. In *Ninth International Conference on Learning Representations 2021.*

Goyal, S.; Kumar, A.; Garg, S.; Kolter, Z.; and Raghunathan, A. 2023. Finetune like you pretrain: Improved finetuning of zero-shot vision models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19338–19347.

Gulati, A.; Qin, J.; Chiu, C.-C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; et al. 2020. Conformer: Convolution-augmented Transformer for Speech Recognition. *Interspeech 2020.*

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Howard, J.; and Ruder, S. 2018. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Ioffe, S.; and Szegedy, C. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, 448–456. pmlr.

Kenton, J. D. M.-W. C.; and Toutanova, L. K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, 2.

Krizhevsky, A.; et al. 2009. Learning multiple layers of features from tiny images.

Kumar, A.; Raghunathan, A.; Jones, R.; Ma, T.; and Liang, P. 2022. Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution. In *International Conference on Learning Representations*.

LeCun, Y.; Bottou, L.; Orr, G. B.; and Müller, K.-R. 2002. Efficient backprop. In *Neural networks: Tricks of the trade*, 9–50. Springer.

Li, D.; and Zhang, H. 2021. Improved regularization and robustness for fine-tuning in neural networks. *Advances in Neural Information Processing Systems*, 34: 27249–27262.

Li, X.; Xiong, H.; Wang, H.; Rao, Y.; Liu, L.; Chen, Z.; and Huan, J. 2019. Delta: Deep learning transfer using feature map with attention for convolutional networks. *arXiv preprint arXiv:1901.09229*.

Lim, H.; Kim, B.; Choo, J.; and Choi, S. 2023. TTN: A domain-shift aware batch normalization in test-time adaptation. *arXiv preprint arXiv:2302.05155*.

Mirza, M. J.; Micorek, J.; Possegger, H.; and Bischof, H. 2022. The norm must go on: Dynamic unsupervised domain adaptation by normalization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 14765–14775.

Peng, X.; Bai, Q.; Xia, X.; Huang, Z.; Saenko, K.; and Wang, B. 2019. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, 1406–1415.

Quinonero-Candela, J.; Sugiyama, M.; Schwaighofer, A.; and Lawrence, N. D. 2008. *Dataset shift in machine learning*. Mit Press.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Recht, B.; Roelofs, R.; Schmidt, L.; and Shankar, V. 2018. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*.

Ren, Y.; Guo, S.; Bae, W.; and Sutherland, D. J. 2023. How to prepare your task head for finetuning. In *The Eleventh International Conference on Learning Representations*.

Santurkar, S.; Tsipras, D.; and Madry, A. 2020. Breeds: Benchmarks for subpopulation shift. *arXiv preprint arXiv:2008.04859*.

Tan, S.; Peng, X.; and Saenko, K. 2020. Class-imbalanced domain adaptation: an empirical odyssey. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, 585–602. Springer.

Trivedi, P.; Koutra, D.; and Thiagarajan, J. J. 2023. A closer look at model adaptation using feature distortion and simplicity bias. *arXiv preprint arXiv:2303.13500*.

Wang, X.; Jin, Y.; Long, M.; Wang, J.; and Jordan, M. I. 2019. Transferable normalization: Towards improving transferability of deep neural networks. *Advances in neural information processing systems*, 32.

Wiesler, S.; and Ney, H. 2011. A convergence analysis of log-linear training. *Advances in Neural Information Processing Systems*, 24.

Xuhong, L.; Grandvalet, Y.; and Davoine, F. 2018. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, 2825–2834. PMLR.

Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R. R.; and Le, Q. V. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Zhang, J. O.; Sax, A.; Zamir, A.; Guibas, L.; and Malik, J. 2020. Side-tuning: a baseline for network adaptation via additive side networks. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, 698–714. Springer.