DR-Tune: Improving Fine-tuning of Pretrained Visual Models by Distribution Regularization with Semantic Calibration

Nan Zhou1,2  Jiaxin Chen2  Di Huang1,2,3*
1State Key Laboratory of Software Development Environment, Beihang University, Beijing, China
2School of Computer Science and Engineering, Beihang University, Beijing, China
3Hangzhou Innovation Institute, Beihang University, Hangzhou, China
{zhounan0431,jiaxinchen,dhuang}@buaa.edu.cn

Abstract

The visual models pretrained on large-scale benchmarks encode general knowledge and prove effective in building more powerful representations for downstream tasks. Most existing approaches follow the fine-tuning paradigm, either by initializing or regularizing the downstream model based on the pretrained one. The former fails to retain the knowledge in the successive fine-tuning phase, thereby prone to be over-fitting, and the latter imposes strong constraints to the weights or feature maps of the downstream model without considering semantic drift, often incurring insufficient optimization. To deal with these issues, we propose a novel fine-tuning framework, namely distribution regularization with semantic calibration (DR-Tune). It employs distribution regularization by enforcing the downstream task head to decrease its classification error on the pretrained feature distribution, which prevents it from over-fitting while enabling sufficient training of downstream encoders. Furthermore, to alleviate the interference by semantic drift, we develop the semantic calibration (SC) module to align the global shape and class centers of the pretrained and downstream feature distributions. Extensive experiments on widely used image classification datasets show that DR-Tune consistently improves the performance when combing with various backbones under different pretraining strategies. Code is available at: https://github.com/weeknan/DR-Tune.

1. Introduction

Nowadays, it has become a prevailing paradigm to pretrain deep models for common use on large-scale datasets and fine-tune them in multiple diverse downstream tasks in the community of computer vision [20, 7]. Due to the data and semantic relevance between pretraining and downstream tasks, the pretrained model implicitly encodes useful prior knowledge, and compared with the ones by training from scratch, it substantially promotes the accuracy of the downstream task and accelerates its training convergence in a variety of applications [21, 45], e.g. image classification, object detection, and semantic segmentation. In particular, when labeled data are quite limited in the downstream task, the issue of over-fitting can be effectively alleviated by using the pretrained model as a training prior.

To facilitate training downstream models with the pretrained ones, many efforts have recently been made. One of the typical ways is to directly take the pretrained model for initialization and fine-tune [24, 58] its weights by elaborately designing task-specific learning objectives [10, 35, 17, 65, 64]. Nevertheless, these methods neglect retaining...
The pretrained prior in the fine-tuning phase and tend to incur the “catastrophic forgetting” problem [40, 6, 16], making the learned model prone to over-fit.

In contrast, another alternative focuses on utilizing the prior knowledge encoded in the pretrained model to regularize the training of downstream models [56, 16]. By introducing extra regularization terms based on a pretrained model either on the weights [56] (see Fig. 1 (a)) or the intermediate feature maps [30, 36] (see Fig. 1 (b)), these methods prevent the downstream model from over-fitting and significantly boost the overall performance; however, they often impose explicit ad-hoc constraints by reducing the discrepancy between the weights or the sample-wise feature maps generated by the pretrained and downstream models, without considering the semantic drift of the pretrained features. As a consequence, they are inclined to suffer from the non-negligible bias caused by the pretrained model, deteriorating the final result which may be even worse than vanilla fine-tuning in specific scenarios as claimed in [10], and leave much room for improvement.

To address the issues above, this paper proposes a novel regularization-based framework for fine-tuning, namely distribution regularization (DR) with semantic calibration (DR-Tune). As Fig. 1 (c) illustrates, different from the existing methods, DR-Tune conducts distribution regularization on the downstream classification head, instead of the encoder. The basic idea behind is to minimize the classification error of the downstream task head according to the pretrained feature distribution in addition to the normally used downstream feature distribution. Unfortunately, the discrepancy between the dynamically updated downstream model and the frozen pretrained model incurs semantic drift between the two distributions as shown in Fig. 2 (c), which hinders the task head from learning correct classification boundaries. To alleviate this drift, we develop the semantic calibration (SC) module to align the pretrained and downstream feature distributions via a holistic rotation matrix as well as a group of class-level translation vectors, which are efficiently estimated by establishing two memory banks. The rotation matrix performs global distance-preserving alignment, while the translation vectors offer the alignment of class center pairs, significantly removing the semantic drift as depicted in Fig. 2 (d).

Intuitively, the proposed DR-Tune framework has two underlying advantages: 1) DR does not impose explicit constraints neither on the weights nor on the intermediate feature maps, largely facilitating optimizing the downstream encoder towards the downstream task; 2) SC greatly reduces the semantic drift and the classification bias is thus alleviated when employing the pretrained feature distribution as regularization, leading to improved fine-tuning results; and 3) as in Fig. 2 (b), by leveraging the extra support from the pretrained feature distribution and the downstream features, the task head benefits generating smoother classification boundaries, restricting the over-fitting risk.

The main contributions are summarized as follows:

1) We propose a novel fine-tuning framework (DR-Tune), which handles over-fitting by regularizing the task-specific head with the pretrained feature distribution.

2) We design the SC module to address the semantic drift between the pretrained and downstream feature distributions, effectively decreasing the bias introduced by the regularization from the pretrained models.

3) We conduct extensive evaluation on popular classification datasets and demonstrate that DR-Tune consistently improves the performance as combined with various network structures under different pretraining schemes.

2. Related Work

2.1. General Model Fine-tuning

Most existing fine-tuning methods focus on downstream tasks by elaborately designing task-specific learning objectives. SCL [17], Bi-tuning [65] and Core-tuning [64] incor-
porate the supervised contrastive loss \cite{29} with the standard cross-entropy (CE) loss, achieving superior performance on classification tasks. M&M \cite{62} improves semantic segmentation by utilizing limited pixel-wise annotations in the downstream dataset in conjunction with the triplet loss. Besides, BSS \cite{10} observes that small eigenvalues incur degradation compared to vanilla fine-tuning, and thus penalizes on the eigenvalues of the learned representation. RIFLE \cite{35} performs fine-tuning by periodically re-initializing the fully connected layers. In general, the methods above neglect retaining the pretrained prior in the fine-tuning phase and tend to over-fit on the downstream task.

In addition, several studies also attempt to apply various adapters \cite{46, 47, 63, 5, 37} or prompts \cite{27, 42, 28, 1} to decrease the computational and storage cost during fine-tuning. Despite their efficiency, these methods sacrifice the performance in accuracy.

### 2.2. Regularization for Model Fine-tuning

Regularization is a prevailing way to make use of the pretrained prior knowledge for fine-tuning. Li et al. \cite{56} apply the $l^2$-norm penalty between the parameters of the pretrained and downstream models, which outperforms the standard weight decay. Yim et al. \cite{57} introduce the knowledge distillation \cite{23, 48} and adopt the distance between the flow of the solution procedure matrix of the pretrained and downstream models as the regularizer. AT \cite{30} and DELTA \cite{36} exploit the attention mechanism and regularize the discrepancy between the intermediate feature maps. \cite{16} assembles multiple distance-based metrics for regularization, which is optimized by the projected gradient descent method. Co-Tuning \cite{59} explores the semantic information of the pretrained dataset and uses the pretrained labels to regularize the fine-tuning process. These methods handle overfitting by imposing explicit ad-hoc constraints to reduce the discrepancy between the weights or sample-wise feature maps of the pretrained and downstream models, but they do not take into account the semantic drift of the pretrained features, thus leaving room for improvement.

Compared to existing solutions as described in Sec. 2.1 and Sec. 2.2, we prevent the downstream model from overfitting by introducing distribution regularization (DR) on the task head. DR leverages the pretrained feature distribution to enforce the task head learning smooth classification boundaries without imposing explicit constraints on backbones, thus facilitating optimizing the downstream encoder. In addition, we observe the semantic drift between the pretrained and downstream feature distributions, and mitigate it by developing a novel semantic calibration (SC) module, which substantially improves the final performance.

### 3. Approach

#### 3.1. Preliminaries

Suppose a pretrained model $g_{\theta_p} \cdot f_{\theta_p}(\cdot)$, where $f_{\theta_p}$ and $g_{\theta_p}$ denote the encoder and the pretraining task head parameterized by $\theta_p$ and $\phi_p$, respectively. Given a set of training data $D = \{(x_i^d, y_i)\}_{i=1}^N$ for the downstream task, we aim to learn a downstream model $g_{\theta_d} \cdot f_{\theta_d}(\cdot)$ by fine-tuning the pretrained model $g_{\theta_p} \cdot f_{\theta_p}(\cdot)$, where $x_i^d$ refers to the $i$-th image with the class label $y_i$, $\theta_d$ and $\phi_d$ are the parameters to be learned for the downstream encoder $f_{\theta_d}$ and the downstream task head $g_{\phi_d}$, respectively.

To learn $\theta_d$ and $\phi_d$, vanilla fine-tuning firstly applies the pretrained parameter $\theta_p$ to initialize $\theta_d$ as $\theta_d(0) := \theta_p$. $\phi_d$ is randomly initialized, which is thereafter jointly learned with $\theta_d$ by optimizing the following objective:

$$ (\theta^*_d, \phi^*_d) = \arg\min_{\theta_d, \phi_d} \mathcal{L}(g_{\phi_d} \cdot f_{\theta_d}; D), $$  

where $\mathcal{L}()$ is the task-specific loss. The fine-tuned model $g_{\phi_d} \cdot f_{\theta_d}$ is used for inference in the downstream task.

Nevertheless, the vanilla fine-tuning strategy is prone to be over-fitting on the downstream data, especially when the training size $N$ is small. To overcome this shortcoming, the regularization-based fine-tuning strategy is employed by introducing a regularization term $\mathcal{R}(\cdot)$ on $\theta_d$ according to $\theta_p$ and optimizing the following objective:

$$ (\theta^*_d, \phi^*_d) = \arg\min_{\theta_d, \phi_d} \mathcal{L}(g_{\phi_d} \cdot f_{\theta_d}; D) + \mathcal{R}(\theta_d; \theta_p). $$

Most of existing fine-tuning methods perform regularization in an ad-hoc manner such as the weight-based ones formulated as $\mathcal{R} = \|\theta_d - \theta_p\|$ as well as the feature-based ones written as $\mathcal{R} = \sum_{i=1}^N \|FM(x_i^d|f_{\theta_d}) - FM(x_i^d|f_{\theta_p})\|$, where $FM(x_i^d|f_{\theta})$ indicates the feature map of $x_i^d$ extracted from the intermediate layer of $f_{\theta}$. The former imposes strong constraints on $\theta_d$, and the later forces the downstream feature $FM(x_i^d)$ to be the same as the pretrained one for each training sample $x_i^d$, both of which impede $\theta_d$ from being sufficiently optimized towards the downstream task.

#### 3.2. Framework Overview

To address the issues above, we propose a novel fine-tuning framework, namely distribution regularization with semantic calibration (DR-Tune).

As illustrated in Fig. 3, given training set $D = \{(x_i^d, y_i)\}$, we extract the downstream representations $\{x_i^d | x_i^d = f_{\theta_d}(x_i^d)\}$ and the pretrained representations $\{x_i^p | x_i^p = f_{\theta_p}(x_i^d)\}$ by the encoders $f_{\theta_d}$ and $f_{\theta_p}$, respectively.

The basic idea of DR-Tune is employing an implicit distribution regularization (DR) $\mathcal{R}_{DR}$$\{\{(z_i^d, y_i)\}|g_{\phi_d}\}$ on the
downstream model, i.e. the task head $g_{\theta^d}$ is enforced to correctly classify the pretrained representations $\{z^p_i\}$, besides the downstream ones $\{z^d_i\}$.

However, as shown in Fig. 2 (c), there exists semantic drift between the pretrained feature distribution and the downstream one. Therefore, directly using $\{z^p_i\}$ for regularization incurs non-negligible bias, thus degrading the performance of the fine-tuned downstream model. To solve this problem, DR-Tune introduces a semantic calibration (SC) module to alleviate the distribution drift. Concretely, as displayed in Fig. 3, DR-Tune employs two queues to build a downstream feature bank $\mathcal{M}^d$ as well as a pretrained feature bank $\mathcal{M}^p$, which are dynamically updated according to the features $\{z^d_i\}$ and $\{z^p_i\}$ in the mini-batch, respectively. $\mathcal{M}^d$ and $\mathcal{M}^p$ efficiently represent the downstream and pretrained feature distribution, based on which the calibration parameters including a global rotation matrix $R$ and a group of class-level translations $\{\delta_i\}_c$ are estimated, where $\delta_i$ is the translation vector for the $c$-th class. During training, the calibrated pretrained features $\{\hat{z}^p_i\}$ are used to form the final distribution regularization as $\mathcal{R}_{\text{DR}}(\{(\hat{z}^p_i, y_i)\}|g_{\theta^d})$. In the testing phase, we skip the SC module as well as the feature banks, and only use the downstream encoder $f_{\theta^d}$ and the head $g_{\theta^d}$ for inference.

The details about the DR term and the SC module are described in Sec. 3.3 and Sec. 3.4, respectively.

3.3. Fine-tuning with Distribution Regularization

In this section, we elaborate the formulation of DR, i.e. $\mathcal{R}_{\text{DR}}(\{(z^p_i, y_i)\}|g_{\theta^d})$.

Formally, suppose the training set $D$ is drawn from the data distribution $\mathcal{X}^d$, the feature distributions of $\{f_{\theta^d}(x^d_i)\}$ and $\{f_{\theta^p}(x^d_i)\}$ are formulated as $\mathcal{Z}^d = P_{\mathcal{X}^d\sim\mathcal{X}^d}(f_{\theta^d}(x))$ and $\mathcal{Z}^p = P_{\mathcal{X}^d\sim\mathcal{X}^d}(f_{\theta^p}(x))$, respectively. It is worth noting that both $\mathcal{Z}^p$ and $\mathcal{Z}^d$ are derived from the same distribution $\mathcal{X}^d$, but by distinct encoders $f_{\theta^p}$ and $f_{\theta^d}$.

Usually, the downstream task-specific learning objective $\mathcal{L}$ can be briefly written as below:

$$\mathcal{L} = - \log Pr_{\mathcal{X}^d\sim\mathcal{X}^d}(\{(z^d_i, y_i)\}|f_{\theta^d}; g_{\theta^d}) ,$$

where $z^d_i = f_{\theta^d}(x^d_i)$ and $Pr_{\mathcal{X}^d\sim\mathcal{X}^d}(\{(z^d_i, y_i)\}|f_{\theta^d}; g_{\theta^d})$ is the joint probability of the training feature set $\{(z^d_i, y_i)\}$ conditioned on $f_{\theta^d}$ and $g_{\theta^d}$.

As aforementioned, $\mathcal{R}_{\text{DR}}$ aims to regularize the task head $g_{\theta^d}$ by enforcing it to classify the pretrained representations $\{z^p_i\}$. To this end, we adopt the following formulation of $\mathcal{R}_{\text{DR}}$

$$\mathcal{R}_{\text{DR}} = - \log Pr_{z^p\sim\mathcal{Z}^p}(\{(z^p_i, y_i)\}|g_{\theta^d}) ,$$

where $y_i$ is the category of $z^p_i$. From Eq. (4), it can be observed that $g_{\theta^d}$ is optimized to maximize the joint probability of $\{(z^p_i, y_i)\}$ when minimizing $\mathcal{R}_{\text{DR}}$, thus forcing $g_{\theta^d}$ to correctly classify $\{z^p_i\}$.

This kind of regularization has the following advantages compared to existing ad-hoc regularizers: 1) $\mathcal{R}_{\text{DR}}$ does not impose any explicit constraints neither on the downstream weights $\theta^d$ nor on the intermediate downstream features, thus bypassing the interference of improper constraints on
fine-tuning $f_{\theta^d}$. 2) As shown in Fig. 2 (b), instead of using the ad-hoc sample-wise regularization, $R_{DR}$ leverages the pretrained feature distribution $Z^p$ for regularization, which explores holistic information to prevent the downstream task head $g_{\phi^d}$ from over-fitting. In the meantime, when combining $R_{DR}$ in Eq. (4) with the task-specific loss $L$ in Eq. (3), as $g_{\phi^d}$ becomes more generalizable, $f_{\theta^d}$ is improved correspondingly. Please refer to the supplementary material for more analysis.

To specify the form of $R_{DR}$, we clarify the joint probability in Eq. (4). By assuming the independent sampling of $(z^p_i, y_i)$, Eq. (4) is rewritten as $R_{DR} = - \sum_{z^p_i \sim Z^p} \log P_r((z^p_i, y_i)|g_{\phi^d})$. For the classification task with $C$ classes, the parameters of $g_{\phi^d}$ can be decomposed as $\phi^d = [\phi_1^d, \phi_2^d, \ldots, \phi_C^d]$, where $\phi_c^d$ corresponds to the ones for the $c$-th class prototype. Similar to the CE loss, given a pretrained sample $(z^p_i, y_i)$, the conditional probability $P_r((z^p_i, y_i)|g_{\phi^d})$ turns to be

$$P_r((z^p_i, y_i)|g_{\phi^d}) = \frac{\exp(\phi_{y_i} \cdot z^p_i)}{\sum_{c=1}^{C} \exp(\phi_c \cdot z^p_i)}.$$ 

Ideally, all pretrained representations $\{z^p_i\}$ of the training set should involve in computation of $R_{DR}$; however it is extremely inefficient to train $g_{\phi^d}$ by using all of them in each iteration. An alternative way is to extract a mini-batch, but it only captures local information of the distribution. Inspired by [53, 20, 52], we make a trade-off by employing a feature bank to approximate the distribution $Z^p$. Specifically, we maintain a queue $M^p = \{v^p_k\}_{k=1}^K$ with a fixed size $K$ by enqueuing the newest features (i.e. the features from a mini-batch), and dequeuing the oldest ones.

Based on $P_r((z^p_i, y_i)|g_{\phi^d})$ and $M^p$, $R_{DR}$ is finally formulated as below:

$$R_{DR} = - \frac{1}{K} \sum_{k=1}^{K} \log \frac{\exp(\phi_{y_k} \cdot v^p_k)}{\sum_{c=1}^{C} \exp(\phi_c \cdot v^p_k)}, \quad (5)$$

As to the task-specific loss for fine-tuning, we adopt the commonly used CE loss:

$$L := L_{CE} = - \frac{1}{B} \sum_{i=1}^{B} \log \frac{\exp(\phi_y \cdot f_{\theta^d}(x^d_i))}{\sum_{c=1}^{C} \exp(\phi_c \cdot f_{\theta^d}(x^d_i))}, \quad (6)$$

where $\{(x^d_i, y_i}\}$ is the mini-batch for computational efficiency, and $B$ is the mini-batch size.

### 3.4. Semantic Calibration

Since the downstream model is dynamically updated during fine-tuning while the pretrained model is kept frozen, the discrepancy between these two models tends to incur a semantic drift between the pretrained feature distribution $Z^p$ and the downstream one $Z^d$ as illustrated in Fig. 2 (c). Ignoring this drift and forcing $g_{\phi^d}$ to classify features from disparate distributions by jointly optimizing $R_{DR}$ in Eq. (5) and $L_{CE}$ in Eq. (6) degrades the performance.

To alleviate the semantic drift, we attempt to estimate a transformation to calibrate $Z^p$ w.r.t. $Z^d$. To overcome the dilemma in balancing the efficiency and accuracy, we maintain a downstream feature bank $M^d = \{v^d_k\}_{k=1}^K$ with size $K$, similar to the pretrained one $M^p = \{v^p_k\}_{k=1}^K$ constructed in the previous section. It is worth noting that $v^p_k$ and $v^d_k$ are two distinct representations for the same image $x_k$.

In practice, the semantic drift between $Z^d$ and $Z^p$ is extremely complicated, and is hard to estimate. In our work, we simplify it by assuming that the drift is mainly caused by a misalignment of global rotation and a set of local ones of the class centers. Accordingly, we calculate a rotation matrix $R$ and the class-level translations $\{\delta_c\}_{c=1}^C$.

In regards of $R$, we estimate it by solving the following optimization problem:

$$R = \arg\min_{R} R' \cdot R = I_d \sum_{k=1}^{K} \| R' \cdot v^d_k - v^p_k \|^2, \quad (7)$$

where $I_d$ is a $d$-dimensional identity matrix.

Eq. (7) can be solved by applying SVD on the covariance matrix between $M^p$ and $M^d$ [39].

As for the class-level translations $\{\delta_c\}_{c=1}^C$, we observe that the inter-class distribution of $Z^p$ is less discriminative due to the lack of supervision in the downstream task. In contrast, $Z^d$ is more competent at distinguishing different classes. Therefore, we maintain $Z^p$ and use the translation transformation to adjust the inter-class distribution of $Z^p$ to be consistent with $Z^d$. More visualization is given in the supplementary material.

With the motivation above, we first estimate the $c$-th class center for $Z^p$ based on $M^p$ as below

$$\mu^p_c = \frac{1}{N_c} \sum_{k=1}^{K} I[y^k_c = c] \cdot R \cdot v^p_k, \quad (8)$$

In Eq. (8), $N_c$ is the number of pretrained features from the $c$-th class, and $I[y^k_c = c]$ is the indicator function, which equals to 1 if $y^k_c = c$ and 0 otherwise.

As for the downstream features, we compute the class center based on $M^d$ in a more elaborative way as follows

$$\mu^d_c = \sum_{k=1}^{K} \alpha_k \cdot I[y^d_k = c] \cdot v^d_k, \quad (9)$$

where the weight

$$\alpha_k = \frac{\exp(\phi_{y^d_k} \cdot v^d_k)}{\sum_{j=1}^{K} I[y^d_j = y^d_k] \cdot \exp(\phi_{y^d_j} \cdot v^d_j)}, \quad (10)$$

represents the confidence of $v^d_k$ that it is correctly classified to its label by the head $g_{\phi^d}$. Since an outlier feature is usu-
Table 1. Comparison of the top-1 accuracy (%) by using various fine-tuning methods based on the self-supervised pretrained model, i.e. ResNet-50 pretrained by MoCo-v2 on ImageNet. ‘*’ indicates that the method is re-implemented. The best results are in bold.

<table>
<thead>
<tr>
<th>Method</th>
<th>ImageNet20</th>
<th>CIFAR10</th>
<th>CIFAR100</th>
<th>DTD</th>
<th>Caltech101</th>
<th>Cars</th>
<th>Pets</th>
<th>Flowers</th>
<th>Aircraft</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-tuning</td>
<td>88.28</td>
<td>94.70</td>
<td>80.27</td>
<td>71.68</td>
<td>91.87</td>
<td>88.61</td>
<td>89.05</td>
<td>98.49</td>
<td>86.87</td>
<td>87.76</td>
</tr>
<tr>
<td>L2SP [56]</td>
<td>88.49</td>
<td>95.14</td>
<td>81.43</td>
<td>72.18</td>
<td>91.98</td>
<td>89.00</td>
<td>89.43</td>
<td>98.66</td>
<td>86.55</td>
<td>88.10</td>
</tr>
<tr>
<td>DELTA [36]</td>
<td>88.35</td>
<td>94.76</td>
<td>80.39</td>
<td>72.23</td>
<td>92.19</td>
<td>88.73</td>
<td>85.94</td>
<td>98.65</td>
<td>87.05</td>
<td>87.99</td>
</tr>
<tr>
<td>M&amp;M [62]</td>
<td>88.53</td>
<td>95.02</td>
<td>80.58</td>
<td>72.43</td>
<td>92.91</td>
<td>88.90</td>
<td>89.60</td>
<td>98.57</td>
<td>87.45</td>
<td>88.22</td>
</tr>
<tr>
<td>BSS [10]</td>
<td>88.34</td>
<td>94.84</td>
<td>80.40</td>
<td>72.22</td>
<td>91.95</td>
<td>88.50</td>
<td>89.50</td>
<td>98.57</td>
<td>87.18</td>
<td>87.94</td>
</tr>
<tr>
<td>RIFLE [35]</td>
<td>89.06</td>
<td>94.71</td>
<td>81.49</td>
<td>72.73</td>
<td>92.84</td>
<td>89.37</td>
<td>87.91</td>
<td>98.65</td>
<td>87.44</td>
<td>88.54</td>
</tr>
<tr>
<td>SCL [17]</td>
<td>89.29</td>
<td>95.33</td>
<td>81.49</td>
<td>72.73</td>
<td>92.84</td>
<td>89.37</td>
<td>87.91</td>
<td>98.65</td>
<td>87.44</td>
<td>88.54</td>
</tr>
<tr>
<td>Bi-tuning [65]</td>
<td>89.06</td>
<td>95.12</td>
<td>81.42</td>
<td>73.53</td>
<td>92.83</td>
<td>89.41</td>
<td>89.90</td>
<td>98.57</td>
<td>87.39</td>
<td>88.58</td>
</tr>
<tr>
<td>Core-tuning [64]</td>
<td>92.73</td>
<td>97.31</td>
<td>84.13</td>
<td>75.37</td>
<td>93.46</td>
<td>90.17</td>
<td>92.36</td>
<td>99.18</td>
<td>89.48</td>
<td>90.47</td>
</tr>
<tr>
<td>SSF* [37]</td>
<td>94.72</td>
<td>95.87</td>
<td>79.57</td>
<td>75.39</td>
<td>90.40</td>
<td>62.22</td>
<td>84.89</td>
<td>92.15</td>
<td>62.38</td>
<td>81.95</td>
</tr>
<tr>
<td>DR-Tune (Ours)</td>
<td>96.03</td>
<td>98.03</td>
<td>85.47</td>
<td>76.65</td>
<td>95.77</td>
<td>90.60</td>
<td>90.57</td>
<td>99.27</td>
<td>89.80</td>
<td>91.35</td>
</tr>
</tbody>
</table>

Table 2. Comparison of the top-1 accuracy (%) by using various fine-tuning methods based on the supervised pretrained model, i.e. ViT-B pretrained on ImageNet. ‘*’ indicates that the method is re-implemented and ‘†’ refers to the training/test split setting as in [61]. The best results are in bold.

<table>
<thead>
<tr>
<th>Method</th>
<th>CIFAR100†</th>
<th>Caltech101†</th>
<th>DTD†</th>
<th>Flowers†</th>
<th>Pets†</th>
<th>SVHN</th>
<th>Sun397</th>
<th>Avg.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linear probing</td>
<td>63.4</td>
<td>85.0</td>
<td>63.2</td>
<td>97.0</td>
<td>86.3</td>
<td>36.6</td>
<td>51.0</td>
<td>68.93</td>
</tr>
<tr>
<td>Adapter [25]</td>
<td>74.1</td>
<td>86.1</td>
<td>63.2</td>
<td>97.7</td>
<td>87.0</td>
<td>34.6</td>
<td>50.8</td>
<td>70.50</td>
</tr>
<tr>
<td>Bias [60]</td>
<td>72.8</td>
<td>87.0</td>
<td>59.2</td>
<td>97.5</td>
<td>85.3</td>
<td>59.9</td>
<td>51.4</td>
<td>73.30</td>
</tr>
<tr>
<td>VPT [27]</td>
<td>78.8</td>
<td>90.8</td>
<td>65.8</td>
<td>98.0</td>
<td>88.3</td>
<td>78.1</td>
<td>49.6</td>
<td>78.49</td>
</tr>
<tr>
<td>SSF [37]</td>
<td>69.0</td>
<td>92.6</td>
<td>75.1</td>
<td>99.4</td>
<td>91.8</td>
<td>90.2</td>
<td>52.9</td>
<td>81.57</td>
</tr>
<tr>
<td>Core-tuning* [64]</td>
<td>66.3</td>
<td>89.7</td>
<td>70.9</td>
<td>99.0</td>
<td>92.3</td>
<td>76.4</td>
<td>52.5</td>
<td>78.16</td>
</tr>
<tr>
<td>DR-Tune (Ours)</td>
<td>81.1</td>
<td>92.8</td>
<td>71.4</td>
<td>99.3</td>
<td>92.4</td>
<td>92.0</td>
<td>54.5</td>
<td>83.36</td>
</tr>
</tbody>
</table>

4. Experimental Results

In this section, we evaluate the performance of DR-Tune by using distinct pretrained models on widely used datasets, compared with the state-of-the-art counterparts.

4.1. Datasets

We evaluate DR-Tune on widely used datasets, including ImageNet20 [12, 26], CIFAR10 & 100 [33], DTD [11], Caltech101 [15], Stanford Cars [32], Oxford Pets [44] & Flowers [43], Aircraft [39], SVHN [41] and Sun397 [54]. Please refer to the supplementary material for more details.

4.2. Details

By following [14, 64], we use ResNet-50 [22] pretrained by MoCo-v2 [8] and ViT-B [13] pretrained in a supervised manner on ImageNet [12] as the backbone in main experiments. Different pretrained strategies and backbones are also evaluated in Sec 4.4. The size (i.e. K) of the memory banks is set as 2,048 by default.

In most of our experiments, we train for 100 epochs by using the SGD optimizer [3] with a cosine decay scheduler, where the weight decay and momentum are fixed as $1 \times 10^{-4}$ and 0.9, respectively. We use the linear decay scheduler on ImageNet20 [26] and the AdamW [38] optimizer to train the ViT [13] backbone. Since the mini-
batch is augmented before the classification head, we set the learning rate of the classification head $1 + \frac{7}{2}$ times that of the backbone. Similar to [31, 7, 64], we utilize random cropping and horizontal flipping for data augmentation with an image size of $224 \times 224$ during training, and center cropping during test.

4.3. Comparison with the State-of-the-art

In the literature, there are mainly two settings for comparison of different methods, i.e., the one based on the self-supervised pretrained model as in [64] and another based on the supervised pretrained model as in [61]. As for the self-supervised setting, we compare our method with the following state-of-the-arts: 1) the baseline method denoted as CE-tuning, which simply uses the pretrained model for initialization and is successively trained on downstream data by the standard CE loss; 2) the regularization-based methods including L2SP [56] and DELTA [36]; 3) other fully fine-tuning methods including M&M [62], BSS [10], RI-FLE [35], Bi-tuning [65], SCL [17] and Core-tuning [64]. As to the supervised setting, the representative parameter efficient methods, including the baseline Linear probing, Adapter [25], Bias [60], VPT [27] and SSF [37], are selected. It is worth noting that the datasets as well as the training/test split used in these two settings are NOT the same; therefore we separately report their results for fair comparison as in Table 1 and Table 2, respectively.

Under the self-supervised pretraining setting, as summarized in Table 1, vanilla fine-tuning (i.e., CE-tuning) performs the worst, indicating the necessity of exploring the pretrained model in downstream tasks, instead of simply using it for initialization. By launching DR on the task head and reducing the semantic drift, DR-Tune largely outperforms the regularization-based methods L2SP and DELTA, promoting their top-1 accuracies averaged by 3.25% and 3.36%, respectively. The other counterparts such as Bi-tuning and Core-tuning focus on designing loss functions to boost the learning of downstream models without the pretrained model for training, thus prone to over-fit. In contrast, DR-Tune applies the pretrained features to facilitate the task head learning smooth classification boundaries and achieves better performance on most datasets. For instance, the accuracy of DR-Tune exceeds the second best Core-tuning by 3.30%/1.34%/2.31% on ImageNet20/CIFAR100/Caltech101 respectively, and is 0.88% higher than Core-tuning on average over all datasets. Under the supervised pretraining setting, as Table 2 shows, our method consistently boosts the averaged top-1 accuracy, promoting the second best method SSF by 1.78%.

Core-tuning and SSF are the most competitive counterparts only under the self-supervised and supervised setting, respectively, and we further re-implement them and evaluate their performance by using the alternative setting, denoted as SSF* and Core-tuning*. As displayed, they fail to retain high performance when using different pretrained models, while our method yields decent results in both the settings, clearly showing its generalizability.

4.4. Generalizability

We further evaluate the generalizability of DR-Tune by combining it with distinct pretraining strategies, backbones as well as the scales of the downstream data.

In regards of different pretraining strategies, except for MoCo-v2 used in Table 1, we integrate DR-Tune with the pretrained models based on the ResNet-50 backbone by: 1) the contrastive self-supervised methods including MoCo-v1 [20], PCL [34], InfoMin [50] and HCSC [18]; 2) the clustering based self-supervised method SwAV [4]; and 3) the prediction based self-supervised method SimSiam [9]. As shown in Table 3, DR-Tune consistently delivers significant improvement on Caltech101 and ImageNet20 compared to CE-tuning, in regardless of the pretraining strategy used.

With respect to distinct backbones, we adopt the widely used residual networks including ResNet(R)-50/101/152 and ResNeXt(RX)-101/152 [55] pretrained by InfoMin [50], as well as the vision transformers including ViT-Base (ViT-B)/Large (ViT-L) [13] pretrained by MAE [19]. As shown in Table 4, DR-Tune obtains gains compared to CE-tuning with distinct backbones. The results on ViT further demonstrate that DR-Tune applies to the Masked Image Modeling pretraining strategy [2].

<table>
<thead>
<tr>
<th>Pretraining Strategy</th>
<th>Caltech101</th>
<th>ImageNet20</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CE-tuning</td>
<td>Ours</td>
</tr>
<tr>
<td>MoCo-v1 [20]</td>
<td>91.18</td>
<td><strong>91.94</strong></td>
</tr>
<tr>
<td>PCL [34]</td>
<td>93.48</td>
<td><strong>94.90</strong></td>
</tr>
<tr>
<td>InfoMin [50]</td>
<td>93.38</td>
<td><strong>95.10</strong></td>
</tr>
<tr>
<td>HCSC [61]</td>
<td>93.89</td>
<td><strong>95.73</strong></td>
</tr>
<tr>
<td>SwAV [4]</td>
<td>92.79</td>
<td><strong>93.94</strong></td>
</tr>
<tr>
<td>SimSiam [9]</td>
<td>82.28</td>
<td><strong>90.33</strong></td>
</tr>
</tbody>
</table>

**Table 3.** Top-1 accuracy (%) of DR-Tune by combining with different pretraining strategies based on ResNet-50, compared to the baseline CE-tuning.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Caltech101</th>
<th>DTD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CE-tuning</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R-50</td>
<td>93.38</td>
<td>68.62</td>
</tr>
<tr>
<td>R-101</td>
<td>94.23</td>
<td><strong>95.10</strong></td>
</tr>
<tr>
<td>R-152</td>
<td>94.48</td>
<td><strong>96.19</strong></td>
</tr>
<tr>
<td>RX-101</td>
<td>94.71</td>
<td><strong>96.39</strong></td>
</tr>
<tr>
<td>RX-152</td>
<td>94.85</td>
<td><strong>96.44</strong></td>
</tr>
<tr>
<td>ViT-B</td>
<td>94.35</td>
<td><strong>96.03</strong></td>
</tr>
<tr>
<td>ViT-L</td>
<td>95.64</td>
<td><strong>97.57</strong></td>
</tr>
</tbody>
</table>

**Table 4.** Top-1 accuracy (%) of DR-Tune by combining with distinct backbones, compared to the baseline CE-tuning.
the centers are more sensitive to hard samples due to small
especially on the fine-grained Cars and Pets datasets, of which
suspicious outlier features, CGA facilitates computing the
promote the performance. By suppressing the weights of
Pets. As demonstrated in Table 7, both GR and CLT clearly
played in Table 6, both DR and SC contribute to the overall
performance. For fine-grained Cars and Flowers, the fea-
ture distributions generated by the pretrained model and the
downstream one exhibit a severe semantic drift, due to their
large discrepancy on the semantic granularity. DR alone
fails to deal with this drift, thus incurring degradation in per-
formance. SC remarkably boosts the overall performance
by mitigating this semantic drift. Please refer to the
supple-
mentary material for more analysis.

Effect of different transformations in SC. The pro-
posed SC module performs feature transformation by a
global rotation (GR) and a group of class-level translations
(CLT) refined by the confidence guided average (CGA).
We therefore evaluate their effects on Caltech101, Cars and
Pets. As demonstrated in Table 7, both GR and CLT clearly
promote the performance. By suppressing the weights of sus-
picious outlier features, CGA facilitates computing the
centers more precisely, further improving the accuracy, es-
specially on the fine-grained Cars and Pets datasets, of which
the centers are more sensitive to hard samples due to small
inter-class discrepancies.

Effect of hyper-parameter. The DR-Tune framework
is hyper-parameter-friendly, and the only hyper-parameter
is the size of the feature banks $K$. Since the learning rate
varies as $K$ changes (see details in Sec. 4.2) in our setting,
we fix it as 0.01 to eliminate its interference. As shown in
Fig. 4, DR-Tune outperforms the baseline by vanilla fine-
tuning and performs steadily with different $K$ values, even
when $K$ is set at a small one (e.g. 64).

5. Conclusion and Limitation
In this paper, we propose a novel framework, namely
distribution regularization with semantic calibration (DR-
Tune), for fine-tuning pretrained visual models on down-
stream tasks. DR-Tune employs DR on the classification
head by leveraging the pretrained feature distribution, and
develops an SC module to alleviate the semantic drift of the
pretrained features relative to the downstream ones. Extensive
comparison results as well as ablation studies on widely
used datasets clearly show the effectiveness and generaliz-
ability of the proposed method.
Despite its merits, DR-Tune has some limitations: 1) It suffers from a high training latency, due to computation of rotations by SVD in SC, which can be further improved by more efficient solutions. 2) SC aligns the downstream and pretrained features by a global feature after average pooling for classification, ignoring spatial misalignment, which is crucial to spatio-sensitive tasks, e.g. object detection and semantic segmentation, leaving room for gains.

Acknowledgment

This work is partly supported by the National Key R&D Program of China (2021ZD0110503), the National Natural Science Foundation of China (62022011 and 62202034), the Research Program of State Key Laboratory of Software Development Environment, and the Fundamental Research Funds for the Central Universities.

References


[19] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In CVPR, 2022.


