Adaptive Random Feature Regularization on Fine-tuning Deep Neural Networks

Shin'ya Yamaguchi* NTT, Kyoto University
Sekitoshi Kanai NTT
Kazuki Adachi NTT
Daiki Chijiwa NTT

Abstract

While fine-tuning is a de facto standard method for training deep neural networks, it still suffers from overfitting when using small target datasets. Previous methods improve fine-tuning performance by maintaining knowledge of the source datasets or introducing regularization terms such as contrastive loss. However, these methods require auxiliary source information (e.g., source labels or datasets) or heavy additional computations. In this paper, we propose a simple method called adaptive random feature regularization (AdaRand). AdaRand helps the feature extractors of training models to adaptively change the distribution of feature vectors for downstream classification tasks without auxiliary source information and with reasonable computation costs. To this end, AdaRand minimizes the gap between feature vectors and random reference vectors that are sampled from class conditional Gaussian distributions. Furthermore, AdaRand dynamically updates the conditional distribution to follow the currently updated feature extractors and balance the distance between classes in feature spaces. Our experiments show that AdaRand outperforms the other fine-tuning regularization requiring auxiliary source information and heavy computation costs.

1. Introduction

Fine-tuning is a standard technique for training deep neural network models. In fine-tuning, we pre-train a model on large-scale source tasks (e.g., ImageNet [28] and WebImageText for CLIP [27]) before training it on target tasks. Fine-tuning can improve the performance and efficiency of training on the target task [12]. However, fine-tuning deep neural networks still suffers from overfitting when small target datasets are used [18].

To alleviate the overfitting, previous studies have proposed regularization terms so that models maintain source knowledge [18–20, 33]. The early methods simply minimize the gap between source and target models on the model parameters [18] or the intermediate outputs [19]. The recent methods have evolved to utilize auxiliary source information such as source classification labels [33] and source datasets [20] to improve the performance. However, the source information is not always available because pre-training methods are not limited to supervised classification and source datasets are often private. For instance, multi-modal pre-training by CLIP [27] provides only pre-trained parameters but not the source dataset (WebImageText). Thus, we cannot directly apply the regularization methods using the auxiliary source information to CLIP pre-trained models.

An alternative regularization strategy that does not require source information is to modify feature vectors or task heads to be desirable for solving downstream classification tasks [10, 36–38]. These methods penalize the feature extractors of fine-tuning models by ℓ_1/ℓ_2 loss for selecting features to train effectively [10, 37] or contrastive loss for obtaining instance discriminative features [36]. More recently, Zhou et al. [38] have shown a regularization method that enforces linear classification head parameters to classify the features extracted from fixed source models. However, this method requires additional memory to store pre-trained features and compute their penalty term. This increases the memory and computation costs of fine-tuning and the complexity of implementation.

To achieve high performance without auxiliary source information and non-negligible computation costs, we spotlight the method proposed by Zhong and Maki [37], which we call random feature regularization (RandReg). RandReg penalizes the feature extractor by the ℓ_2 distance between feature vectors and random reference vectors. The reference vectors are drawn from a class-agnostic prior distribution that is independent of the target task, e.g., uniform distribution. By perturbing features with randomness in addition to penalizing the feature norms, RandReg boosts fine-tuning performance without auxiliary source information and heavy additional computation costs.

Although RandReg boosts the performance, we empirically found that (a) the performance gain of RandReg depends on the choice of the prior distribution because the feature norms and scales of pre-trained models vary widely

*Corresponding author. shinya.yamaguchi@ntt.com

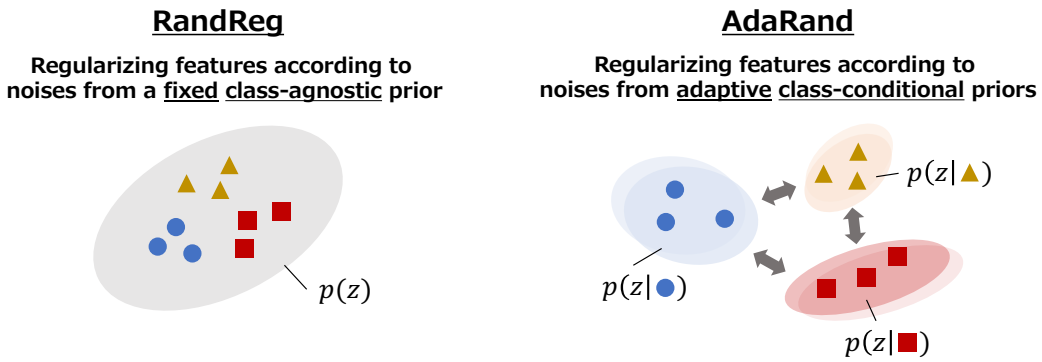


Figure 1. Intuitive comparison of RandReg and AdaRand (proposed method). RandReg regularizes a feature extractor by minimizing the gap between input features and noises generated from a fixed prior distribution. Although RandReg is very simple, it tends to concentrate the features in local regions due to the prior being fixed and class-agnostic, preventing separate classes. In contrast, AdaRand adopts class conditional priors and dynamically updates them with the running feature statistics of training models and the maximization distances between each pair of class conditional priors. This helps models to obtain more separable features and improve accuracy.

depending on the pre-training methods (Table 1), (b) in some cases, RandReg is unexpectedly inferior to the ℓ_2 feature regularization without randomness, (c) RandReg has an unexpected effect of reducing the feature norm and entropy due to the single class-agnostic prior, which leads to limiting gradients of cross-entropy loss and mutual information between features and target labels. Therefore, naïvely introducing RandReg with a simple prior may limit the performance of target models.

To address the challenges of RandReg, we propose *adaptive random feature regularization* (AdaRand), which extends RandReg to be effective for arbitrary pre-training methods including self-supervised learning and CLIP. AdaRand uses a parametric class conditional Gaussian prior that is dynamically updated during fine-tuning instead of a fixed class-agnostic prior. By initializing the prior distribution with the statistics of feature vectors computed on pre-trained models for each target class, AdaRand performs regularization stably without suffering from the differences in features due to the choice of pre-training methods. Whereas RandReg causes small feature norms and entropy, AdaRand prevents them by dynamically updating the prior parameters of each class according to the fine-tuning process. The objective function consists of (i) fitting the mean parameters to the class-wise running mean of feature vectors during fine-tuning and (ii) penalizing them so that they are not similar to any other class. That is, the prior distributions are moved toward the distribution of the current feature vectors while maintaining a margin between classes. This improves the mutual information between features and target labels, resulting in separable clusters of feature vectors that are suitable for the target classification task (Figure 1).

We conduct experiments to evaluate AdaRand with various (classification, self-supervised, and CLIP) pre-training methods on multiple datasets. The experiments show that AdaRand outperforms RandReg and existing fine-tuning methods depending on auxiliary source information and non-negligible computation costs, even though AdaRand does not require either.

2. Related Work

Many regularization methods for fine-tuning deep neural networks are based on the assumption that maintaining source knowledge is beneficial for solving target tasks. According to this assumption, Li et al. [18] have presented a simple regularization called L2SP, which minimizes the parameters between source and target models during fine-tuning. A subsequent study [19] has shown that learning to maximize the similarity between the output feature vectors of source and target models can outperform L2SP. To prevent catastrophic forgetting and negative transfer, batch spectral shrinking (BSS, [5]) penalizes smaller singular values of the batch feature matrices. Although these methods are simple and flexible for arbitrary pre-training methods, the performance improvements are limited. To achieve more practical performance, Co-tuning [33] leverages source knowledge contained in the source task-specific layers on the head of pre-trained models, which are often discarded during fine-tuning. Specifically, in addition to target tasks, Co-tuning simultaneously solves a pseudo-source task that is defined by soft-source labels corresponding to each target label. BTfW [9] and UOT [21] search target-related subsets of the source dataset through the selection algorithm and train a model on both the target and target-related source subset to directly transfer source knowledge. By leveraging auxiliary source information (i.e., source class labels and datasets), Co-tuning and UOT have achieved impressive fine-tuning performance on target tasks. However, since recent powerful

pre-training models such as CLIP [27] are often not trained on classification or publicly available datasets, we cannot apply the previous methods to them.

On the other hand, there are regularization methods that refine parameters or features without explicitly maintaining source knowledge. Takada and Fujisawa [29] have shown that the ℓ_1 regularization on training parameters helps to select the parameters to be updated and improve fine-tuning performance. Hariharan and Girshick [10] have proposed the ℓ_1/ℓ_2 regularization methods on feature vectors called feature norm penalty (FNP), which restrict feature activation to extract only useful information with limited volume target data. Subsequently, Zhong and Maki [37] have proposed RandReg, which minimizes the gap between feature vectors and the random reference vectors from a uniform prior distribution. RandReg can be regarded as an advanced method of FNP because the definition of RandReg is decomposed by the ℓ_1/ℓ_2 regularization term and perturbation term, which is designed to help models not to be trapped in local minima [37]. Our work is positioned as one of the regularization methods without the assumption of auxiliary source information and improves RandReg by introducing class conditional Gaussian priors that are dynamically updated to prevent the small norm and low entropy features.

The assumption of conditional Gaussian (mixture) distributions over the feature spaces is often used in the context of adversarial robustness [25, 30]. In contrast, our method aims to improve the performance of fine-tuning by regularizing the feature extractor with the reference vectors sampled from the conditional distributions.

3. Preliminary

3.1. Problem Setting

In this paper, we consider a standard problem of fine-tuning deep neural networks on a K class classification task. We train a neural network model $f_\theta : \mathcal{X} \to \mathcal{Y}$ on a labeled target dataset $\mathcal{D} = \{(x^i, y^i) \in \mathcal{X} \times \mathcal{Y}\}_{i=1}^N$, where $\mathcal{X}$ and $\mathcal{Y}$ are the input and output label spaces, respectively. f_θ is defined by a composition of a feature extractor $g_\phi : \mathcal{X} \to \mathbb{R}^d$ and a weight matrix for linear classification $W \in \mathbb{R}^{d\times K}$, i.e., $f_\theta = W^\top g_\phi$ and $\theta = [\phi, W]$. Here, θ is initialized by $\theta_\mathrm{s} = [\phi_\mathrm{s}, W_\mathrm{s}]$, which is pre-trained on large-scale source datasets through arbitrary pre-training methods such as supervised training [32], self-supervised contrastive training [4, 13], and multimodal training [27].

3.2. Random Feature Regularization

We recall the principle of random feature regularization (RandReg) [37]. RandReg is a regularization method for fine-tuning deep neural networks that uses a prior distribution $p(z)$ of the random reference vector $z \in \mathbb{R}^d$, i.e., $z \sim p(z)$. The objective function is defined as follows:

$$\min_{\theta=[\phi,W]} \mathcal{L}_\mathrm{cls}(\theta) + \lambda \mathcal{L}_\mathrm{reg}(\phi), \tag{1}$$

$$\mathcal{L}_\mathrm{cls}(\theta) = \mathbb{E}_{(x,y)\in\mathcal{D}}\, \ell_\mathrm{CE}(f_\theta(x), y), \tag{2}$$

$$\mathcal{L}_\mathrm{reg}(\phi) = \mathbb{E}_{x\in\mathcal{D}}\, \|g_\phi(x) - z\|_2^2, \tag{3}$$

where ℓ_CE is cross-entropy loss. Intuitively, by minimizing the gap between $g_\phi(x)$ and z, $\mathcal{L}_\mathrm{reg}(\phi)$ makes the feature extractor g_ϕ forms the output feature vectors according to the prior distribution $p(z)$. The original paper [37] explains that RandReg improves classifiers because the reference vector $\boldsymbol{z}$ enlarges the variance of the gradient and prevents the model from overfitting. However, we found that the performance gain of RandReg largely depends on the combinations of pre-training methods and priors, and the naïve randomness by RandReg is not effective in some cases.

4. Observations of RandReg

In this section, we analyze RandReg through preliminary experiments from the perspective of pre-training methods and priors. We evaluated RandReg with pre-trained ResNet-50 [11] models. We provide more detailed training settings in Sec. 6.1. In summary, we obtained three observations.

- Effective prior distribution depends on pre-training methods and the features of the pre-trained models.
- RandReg underperforms the ℓ_2 feature regularization without randomness (i.e., FNP [10]), indicating that there are cases where the naïve randomness is not effective.
- RandReg makes training models generate features with small norms and less diversity.

Consequently, based on these observations, we discuss the challenges of RandReg in terms of fixed prior distributions and features with small norms and less diversity.

4.1. Effects of Prior Distribution

We examine the performance when varying prior distributions in RandReg. We tried three priors for training target models: **uniform distribution** $U(0, 1)$, **standard Gaussian distribution** $\mathcal{N}(0, I)$, and **Gaussian distribution with pre-computed statistics** $\mathcal{N}(\mu_\mathrm{s}, \sigma_\mathrm{s}^2 I)$, which is parameterized by feature mean $\mu_\mathrm{s} \in \mathbb{R}^d$ and variance $\sigma_\mathrm{s}^2 \in \mathbb{R}^d$ computed with pre-trained weights on the target dataset. To assess the importance of the randomness in RandReg, we also tested the ℓ_2 feature norm regularization without randomness (FNP, [10]), which is equivalent to the case where z is 0 in Eq. (3). We evaluated these priors on multiple pre-trained models with four pre-training methods including classification [1], SimCLR [4], Barlow Twins [34], and CLIP [27]. Table 1 shows the top-1 test accuracy on the Cars [15] dataset. We observed that RandReg improves the fine-tuning baselines for all pre-trained models, but the best prior depends on pre-trained methods. Table 1 also lists the averaged feature

Table 1. Analysis of random feature regularization (RandReg [37]) in top-1 test accuracy on various pre-training methods (Cars, ResNet-50). We also report the statistics of target data computed on a pre-trained model: the averaged feature norms ($\|g_{\phi_s}(x)\|$) and the dimension-wise averaged mean and variance ($(\bar{\mu}_s = \sum_{i=1}^{d} \mu_s[i], \bar{\sigma}_s^2 = \sum_{i=1}^{d} \sigma_s^2[i])$). The performance depends on the combinations of pre-training methods and prior. Moreover, there is a case that the accuracy of RandReg degrades from simple ℓ_2 feature regularization (i.e., FNP [10]).

Pre-trained Method	$\|g_{\phi_s}(x)\|_2^2$	$(\bar{\mu}_s, \bar{\sigma}_s^2)$	Fine-tuning	FNP [10]	RandReg-$U(0,1)$	RandReg-$\mathcal{N}(0,1)$	RandReg-$\mathcal{N}(\mu_s, \sigma_s^2)$
ImageNet Classification	19.58	$(4.18\times10^{-1}, 2.69\times10^{-1})$	$89.14^{\pm.42}$	$90.27^{\pm.10}$	$90.59^{\pm.24}$	$90.42^{\pm.12}$	$\mathbf{90.61}^{\pm.24}$
ImageNet SimCLR [4]	1.34	$(2.60\times10^{-2}, 3.97\times10^{-2})$	$83.73^{\pm.73}$	$\mathbf{84.53}^{\pm.32}$	$84.08^{\pm.21}$	$84.03^{\pm.07}$	$83.91^{\pm.04}$
ImageNet Barlow Twins [34]	3.67	$(5.53\times10^{-2}, 6.94\times10^{-2})$	$86.98^{\pm.16}$	$87.44^{\pm.15}$	$87.24^{\pm.30}$	$\mathbf{87.74}^{\pm.33}$	$87.65^{\pm.48}$
CLIP [27]	11.70	$(5.92\times10^{-4}, 3.12\times10^{-2})$	$88.72^{\pm.24}$	$89.96^{\pm.05}$	$90.19^{\pm.40}$	$90.59^{\pm.24}$	$\mathbf{90.78}^{\pm.07}$

norms ($\|g_{\phi_s}(x)\|$) and the dimension-wise averaged mean and variance ($\bar{\mu}_s = \sum_{i=1}^{d} \mu_s[i], \bar{\sigma}_s^2 = \sum_{i=1}^{d} \sigma_s^2[i]$) that are computed on each pre-trained feature extractor by forwarding target data before fine-tuning. Uniform and standard Gaussian distributions are effective when the feature scale is small (e.g., Barlow Twins), but when the scale is large (e.g., ImageNet Classification), Gaussian distributions with pre-computed statistics, i.e., $\mathcal{N}(\mu_s, \sigma_s^2 I)$, are the best. This implies that considering a gap between the feature distributions of pre-trained models and prior distributions is important in selecting priors. However, RandReg is inferior to FNP in the case of SimCLR, where the feature norm and scale are quite small, even when we use pre-computed statistics. This means that the naïve randomness introduced by RandReg is not effective for fine-tuning in some cases.

4.2. Effects on Feature Norm and Diversity

Next, we investigate the effects on features caused by RandReg. Here, we focus on the feature norm and diversity. This is because the regularization term of RandReg in Eq. (3) directly affects the feature norm $\|g_\phi(x)\|_2^2$ by the squared loss, and it can restrict the diversity of the features by the prior $p(z)$. We interpret the feature diversity as the feature entropy $H(g_\phi(x))$, and estimate $H(g_\phi(x))$ by the differential entropy estimator with the assumption that the probabilistic density of $g_\phi(x)$ is constant in an ϵ-ball around a feature $g_\phi(x^i)$ for a randomly sampled x^i [8]:

$$H(g_\phi(x)) \approx \frac{d}{N(N-1)} \sum_{i \neq j} \log \|g_\phi(x^i) - g_\phi(x^j)\|_2^2, \quad (4)$$

where d is the dimension of a feature vector and N is the dataset size. Figure 2 plots the feature norm $\|g_\phi(x)\|_2^2$ and entropy $H(g_\phi(x))$ for each epoch in training; we calculated $\|g_\phi(x)\|_2^2$ and $H(g_\phi(x))$ on training samples. While fine-tuning slightly increases $\|g_\phi(x)\|_2^2$ and $H(g_\phi(x))$, RandReg gradually decreases both of them, which means RandReg produces small feature vectors with less diversity.

4.3. Challenges of RandReg

Fixed prior distribution. From the results in Sec. 4.1, although RandReg can improve fine-tuning regardless of pre-training methods, we should select an effective prior distribution to adjust the reference vectors to the pre-trained feature extractor and obtain the best performance. This is challenging in practice because it is not obvious which prior distribution is the best choice for the pre-training method and the manual search is costly; naïvely using pre-computed statistics μ_s and σ_s^2 did not always achieve the best performance in Table 1. If the prior is not effective, there is a risk that the randomness will not work effectively as in the case of SimCLR in Table 1. Thus, we need to efficiently search for appropriate prior distributions in fine-tuning.

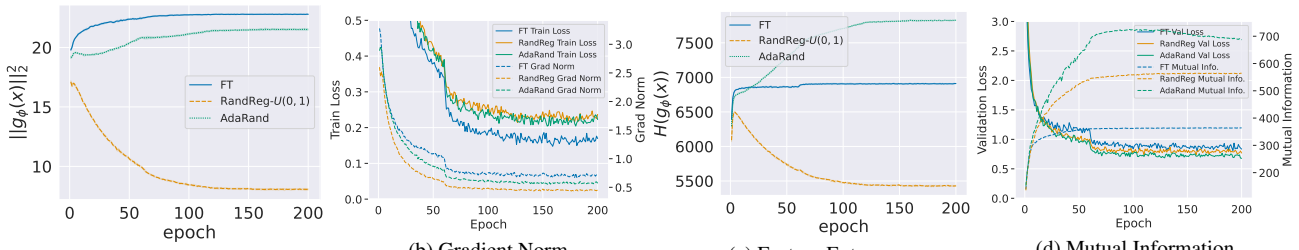
Small norms and less diversity. The experiments in Sec. 4.2 show that RandReg degenerates the feature norm $\|g_\phi(x)\|_2^2$ and entropy $H(g_\phi(x))$. This poses potential challenges for fine-tuning. First, the small feature norm may vanish the gradient of target loss functions. In a classification task, the gradient of the cross-entropy loss with respect to W of the classifier is formulated as follows [10].

$$\|\nabla_W \ell_{\mathrm{CE}}(f_\theta(x), y)\|_2^2 = \sum_{k=1}^{K} \left(f_\theta(x)[k] - \delta_{yk}\right)^2 \|g_\phi(x)\|_2^2, \quad (5)$$

where $f_\theta(x)[k] = W^\top g_\phi(x)[k]$ is the output of the classifier for the k-th class that is normalized by softmax function, and δ_{yk} is 1 if $y = k$ otherwise 0. Since Eq. (5) contains the product of $\|g_\phi(x)\|_2^2$, degenerating $\|g_\phi(x)\|_2^2$ leads to vanishing $\|\nabla_W \ell_{\mathrm{CE}}\|_2^2$, resulting in stagnation of fine-tuning. Second, the low entropy limits the model's ability to learn effective feature representations for solving the target task in terms of mutual information, which is strongly related to model performance [14]. Mutual information $I(g_\phi(x); y)$ measures the amount of shared information (i.e., correlation) between the feature vector $g_\phi(x)$ and label y as defined by

$$I(g_\phi(x); y) = H(g_\phi(x)) - H(g_\phi(x)|y). \quad (6)$$

The first term on the right-hand side $H(g_\phi(x))$ can be interpreted as the "diversity" of features and the second term $-H(g_\phi(x)|y)$ as the "tightness" for each class [3, 35]. In this sense, RandReg limits the mutual information of the feature representations by decreasing the diversity term $H(g_\phi(x))$. Although RandReg can increase the mutual information than the baseline as shown in Fig. 2d, there is a potential to learn a more effective feature representation if we resolve the decrease of $H(g_\phi(x))$.



(a) Feature Norm (b) Gradient Norm (c) Feature Entropy (d) Mutual Information

Figure 2. Statistics on feature vectors during training (Cars, ResNet-50 pre-trained with ImageNet classification). RandReg tends to decrease the feature norm $||g_\phi(x)||_2^2$ and entropy $H(g_\phi(x))$. This implicitly limits the gradient norm of cross-entropy loss $\|\nabla_W \ell_{\text{CE}}(f_\theta(x), y)\|_2^2$ and the discriminability of features represented by mutual information $I(g_\phi(x); y)$. In contrast, our AdaRand prevents the models from decreasing $||g_\phi(x)||_2^2$ and $H(g_\phi(x))$, and thus achieves larger $\|\nabla_W \ell_{\text{CE}}(f_\theta(x), y)\|_2^2$ and $I(g_\phi(x); y)$ than RandReg.

Algorithm 1 AdaRand

Require: Training dataset $\mathcal{D}$, target model f_θ, training batchsize B, step size η and ξ, trade-off parameter λ
Ensure: Trained classifier f_θ

Initialize $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ for conditional priors by Eq. (9) and (10).
$\bar{\boldsymbol{\mu}} \leftarrow \boldsymbol{\mu}$
while not converged **do**
  $\{(x^i, y^i)\}_{i=1}^B \sim \mathcal{D}$
  $\{z^i | z^i \sim \mathcal{N}(\mu_{y^i}, \sigma_{y^i}^2)\}_{i=1}^B$
  // Updating $f_\theta = W^\top g_\phi$
  $\theta \leftarrow \theta - \frac{\eta}{B} \sum_{i=1}^B \nabla_\theta (\ell_{\text{CE}}(f_\theta(x^i), y^i) + \lambda \|g_\phi(x^i) - z^i\|_2^2)$
  // Updating conditional priors
  $\bar{\boldsymbol{\mu}} \leftarrow \text{ema\_update}(\{(g_\phi(x^i), y^i)\}_{i=1}^B)$ // Eq. (14),(15)
  $\boldsymbol{\mu} \leftarrow \boldsymbol{\mu} - \xi \nabla_{\boldsymbol{\mu}} (\ell_{\text{intra}}(\boldsymbol{\mu}, \bar{\boldsymbol{\mu}}) + \ell_{\text{inter}}(\boldsymbol{\mu}))$ // Eq. (11)-(13)
end while

5. Proposed Method

We propose adaptive random feature regularization (AdaRand) by extending RandReg in terms of prior distributions. AdaRand adopts adaptive class conditional priors, which automatically search for prior parameters and prevent feature vectors from degenerating the norm and entropy (Fig. 1). The overall procedure of AdaRand is described in Algorithm 1. Before fine-tuning, we initialize the conditional priors by computing the mean and variance of feature vectors of target data on pre-trained models. During fine-tuning, we regularize target models by the reference vectors from the conditional priors and then update the mean parameters of the prior by approaching them to the running feature mean on training models (ℓ_{intra}) and maximizing the distances between classes (ℓ_{inter}).

5.1. Conditional Prior

Overview. To overcome the challenge of the less diverse feature vectors in RandReg, we introduce conditional prior distributions for generating the reference vectors. As illustrated in Fig. 1, RandReg uses class-agnostic prior such as a uniform distribution. This prevents models from naturally enlarging the marginal entropy $H(g_\phi(x))$ to classify labels of input data. In order to achieve high performance in target classification tasks, the clusters of feature vectors should be separated for each class. In this sense, the reference vector for the regularization term should be conditioned by class labels of input data. Then, the prior of AdaRand can be regarded as the marginal distribution over class labels.

Definition. The prior distribution is defined as follows.

$$p(z) = \sum_{k=1}^{K} p(z|y_k) p(y_k), \tag{7}$$

where $p(z|y_k) = \mathcal{N}(\mu_k, \sigma_k^2 I)$. We formalize $p(z)$ as a mixture of diagonal Gaussian distributions because the feature vectors of softmax neural classifiers are known to follow class conditional Gaussian distributions [2, 17]. When sampling the reference vectors, we generate a random reference vector z^i from the class label y^i of the input x^i, i.e.,

$$z^i \sim \mathcal{N}(\mu_{y^i}, \sigma_{y^i}^2 I). \tag{8}$$

Using these conditional reference vectors in Eq. (3), we can expect that the regularization effect of random noise can be enjoyed while avoiding the decrease of $H(g_\phi(x))$ due to class-agnostic prior distributions.

Initialization. For the prior distribution, we initialize μ_k and σ_k^2 to the statistics of the pre-trained model and update them adaptively. As discussed in Sec. 4.3, the hyperparameter searches for the initial parameters are challenging. Therefore, we initialize μ_k and σ_k^2 with the class-wise mean and variance of feature vectors computed on target data through pre-trained models as

$$\mu_k = \frac{1}{N_k} \sum_{i=1}^{N_k} g_{\phi_{\text{s}}}(x^i), \tag{9}$$

$$\sigma_k^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} (g_{\phi_{\text{s}}}(x^i) - \mu_k)^2, \tag{10}$$

where N_k is the number of samples labeled as a class k and $g_{\phi_{\text{s}}}$ is the pre-trained feature extractor before fine-tuning. For simple notation in the latter sections, we denote the sets of mean and variance parameters as $\boldsymbol{\mu} = \{\mu_k\}_{k=1}^K$ and $\boldsymbol{\sigma}^2 = \{\sigma_k^2\}_{k=1}^K$.

5.2. Adaptive Prior Update

Overview. During fine-tuning, we adaptively optimize the prior parameter set $\boldsymbol{\mu}$ to adjust the reference vectors according to the training progress of the feature extractors. This addresses the challenges of fixed prior distributions and small feature norms of RandReg. That is, we aim to avoid producing small feature norms without manually searching for prior parameters. Furthermore, we penalize $\boldsymbol{\mu}$ so that each μ_k is independent to the other class parameters $\{\mu_l\}_{l\neq k}$. This facilitates models to form tight class conditional feature clusters, corresponding to maximizing the tightness term $-H(g_\phi(x)|y)$ of the mutual information given by Eq. (6).

Objective Function. The objective function for updating $\boldsymbol{\mu}$ is defined as follows.

$$\mathcal{L}_{\text{ada}} = \ell_{\text{intra}}(\boldsymbol{\mu}, \bar{\boldsymbol{\mu}}) + \ell_{\text{inter}}(\boldsymbol{\mu}), \tag{11}$$

$$\ell_{\text{intra}}(\boldsymbol{\mu}, \bar{\boldsymbol{\mu}}) = \tfrac{1}{K}\textstyle\sum_{k=1}^{K} D(\mu_k, \bar{\mu}_k), \tag{12}$$

$$\ell_{\text{inter}}(\boldsymbol{\mu}) = -\tfrac{1}{K(K-1)}\textstyle\sum_{k=1}^{K}\sum_{l\neq k}^{K} D(\mu_k, \mu_l), \tag{13}$$

where, $\bar{\boldsymbol{\mu}}$ is the running mean vectors of $g_\phi(x)$ in training, $D(\cdot)$ is a distance function such as cosine distance, i.e., $1-\frac{\mu_k\cdot\bar{\mu}_k}{\|\mu_k\|_2\|\bar{\mu}_k\|_2}$. We update $\bar{\boldsymbol{\mu}}$ by exponential moving average (EMA) with training samples in batch for each step as

$$\bar{\mu}_k \leftarrow \alpha\bar{\mu}_k + (1-\alpha)\hat{\mu}_k, \tag{14}$$

$$\hat{\mu}_k = \tfrac{1}{B_k}\textstyle\sum_{i=1}^{B_k} g_\phi(x^i), \tag{15}$$

where α is a parameter for controlling the decay of the past information. We fix $\alpha = 0.5$ throughout this paper; we evaluate the effect of α in the supplement. If there are no samples belonging to a class k in the batch, the EMA update will be skipped. For $D(\cdot)$, we used the cosine distance in our experiments because it achieved the best performance empirically. Intuitively, ℓ_{intra} makes μ_k approach the feature region that the model is currently learning by fine-tuning, and encourages the model to focus on learning the current region through the reference random vectors. Meanwhile, ℓ_{inter} corresponds to maximizing $H(g_\phi(x))$ because the k-th class mean parameter μ_k is penalized for moving away from other class parameters $\{\mu_l\}_{l\neq k}$. Then, the class-wise reference vectors help to gather features for each class (i.e., maximizing $-H(g_\phi(x)|y)$).

6. Experiments

We evaluate AdaRand on the combinations of six visual classification tasks, four pre-training methods, and three neural network architectures. Furthermore, we conduct qualitative and quantitative experiments to assess the feature space through PCA visualization, feature norms, loss gradients, and mutual information. We also show the other experiments, including fine-tuning CLIP on ImageNet, in the supplementary.

6.1. Settings

Baselines w/o source information. We compare AdaRand with the following baselines. **Fine-tuning (FT)**: training f_θ with pre-trained weight θ_{s}. **FNP** [10]: fine-tuning ℓ_2 penalty on $g_\phi(x)$, i.e., $\|g_\phi(x)\|_2^2$. **L2SP** [18]: fine-tuning f_θ with an ℓ_2 penalty on θ not to diverge from θ_{s}, i.e., $\|\theta-\theta_{\text{s}}\|_2^2$. **DELTA** [19]: fine-tuning f_θ with a penalty on the intermediate output not to diverge from one of θ_{s}. **BSS** [5]: fine-tuning f_θ by penalizing the singular values of the batch feature matrices. **RandReg** [37]: fine-tuning f_θ with a penalty term to minimize the gap between feature vectors $g_\phi(x)$ and reference vectors $z\sim p(z)$, i.e., $\|g_\phi(x)-z\|_2^2$. **Core-tuning** [36]: fine-tuning f_θ with supervised focal contrastive loss. **DR-Tune** [38]: fine-tuning f_θ by penalizing W to classify the features extracted from source models into target classes.

Baselines w/ source information. To assess the practicality of AdaRand, we additionally used the following baseline methods requiring auxiliary source information. **Co-Tuning** [33]: fine-tuning f_θ with simultaneous pseudo source tasks defined by mapping target and source class labels. **UOT** [21]: fine-tuning f_θ with simultaneous partial source tasks by extracting source samples related to target tasks with optimal-transport-based mapping algorithm.

Datasets. We used six image datasets for classification tasks in various domains: **Aircraft** [23], **Birds** [31], **Cars** [15], **DTD** [6], **Flowers** [24], and **Pets** [26]. Furthermore, we reduced the Cars dataset by $\{10, 25, 50\}\%$ in volume on a fixed random seed to evaluate smaller dataset cases. We randomly split a dataset into $9:1$ and used the former as the training set and the latter as the validation set.

Architectures. We basically used the **ResNet-50** architecture [11]. To confirm the flexibility among architectures, we also evaluated our method with **ViT-B** [7].

Training. As the pre-training methods, we used supervised classification [32] and self-supervised pre-training (SimCLR [4] and Barlow Twins [34]) on ImageNet. Further, we used the pre-trained weights of CLIP [27]. For training on downstream classification tasks with ResNet-50, we trained f_θ by the Nesterov momentum SGD for 200 epochs with a momentum of 0.9, and an initial learning rate of 0.01; we decayed the learning rate by 0.1 at 60, 120, and 160 epochs. For ViT-B, we used the AdamW [22] optimizer with the initial learning rate of 3.0×10^{-5} that decayed by cosine annealing. We used mini-batch sizes of 64. The input samples were resized into a resolution of 224×224. We used λ of 1.0; we discuss the effect of λ in the supplement. We selected the final model by checking the validation accuracy for each epoch. We implemented the training and evaluation with PyTorch-1.11. We ran the experiments three times on a 24-core Intel Xeon CPU with an NVIDIA A100

Table 2. Top-1 test accuracy (%) of various combinations of pre-training methods and neural network architectures (Cars).

Pre-training Method	Architecture	Fine-tuning	FNP [10]	DR-Tune [38]	RandReg-Best	AdaRand (Ours)
ImageNet Classification	RN-50	$89.14^{\pm.42}$	$90.27^{\pm.10}$	$90.38^{\pm.59}$	$90.61^{\pm.24}$	$\mathbf{91.17}^{\pm.13}$
ImageNet Classification	ViT-B/32	$78.56^{\pm1.3}$	$81.77^{\pm.21}$	$79.49^{\pm.51}$	$82.46^{\pm.20}$	$\mathbf{83.84}^{\pm.13}$
ImageNet Classification	ViT-B/16	$87.35^{\pm.53}$	$88.75^{\pm.44}$	$88.19^{\pm.26}$	$88.88^{\pm.26}$	$\mathbf{89.54}^{\pm.17}$
ImageNet SimCLR [4]	RN-50	$83.73^{\pm.73}$	$84.53^{\pm.32}$	$84.05^{\pm.17}$	$84.08^{\pm.21}$	$\mathbf{85.51}^{\pm.05}$
ImageNet Barlow Twins [34]	RN-50	$86.98^{\pm.16}$	$87.44^{\pm.15}$	$86.69^{\pm.23}$	$87.74^{\pm.33}$	$\mathbf{88.23}^{\pm.39}$
CLIP [27]	RN-50	$88.72^{\pm.24}$	$90.19^{\pm.40}$	$90.16^{\pm.22}$	$90.78^{\pm.07}$	$\mathbf{91.25}^{\pm.63}$
CLIP [27]	ViT-B/32	$83.56^{\pm.60}$	$85.79^{\pm.52}$	$85.71^{\pm.59}$	$86.83^{\pm.34}$	$\mathbf{87.40}^{\pm.48}$
CLIP [27]	ViT-B/16	$90.35^{\pm.23}$	$91.24^{\pm.03}$	$90.47^{\pm.26}$	$91.33^{\pm.44}$	$\mathbf{92.84}^{\pm.48}$

Table 3. Top-1 test accuracy (%) on multiple datasets (ResNet-50 pre-trained with ImageNet classification).

Method / Dataset	Aircraft	Birds	Cars	DTD	Flower	Pets
Fine-tuning	$67.78^{\pm.09}$	$74.80^{\pm.56}$	$88.29^{\pm.12}$	$63.88^{\pm.31}$	$94.58^{\pm.21}$	$89.45^{\pm.18}$
FNP [10]	$70.27^{\pm.42}$	$79.57^{\pm.41}$	$90.27^{\pm.10}$	$72.13^{\pm.04}$	$95.07^{\pm.06}$	$91.21^{\pm.12}$
DR-Tune [38]	$73.89^{\pm.10}$	$76.63^{\pm.31}$	$90.38^{\pm.59}$	$70.55^{\pm.28}$	$93.43^{\pm.17}$	$\mathbf{93.24}^{\pm.11}$
RandReg-$U(0,1)$	$71.65^{\pm.59}$	$79.82^{\pm.21}$	$90.42^{\pm.24}$	$72.32^{\pm.41}$	$96.02^{\pm.05}$	$91.72^{\pm.11}$
RandReg-$\mathcal{N}(0,1)$	$73.11^{\pm.88}$	$80.48^{\pm.06}$	$90.32^{\pm.41}$	$72.57^{\pm.26}$	$95.01^{\pm.11}$	$91.80^{\pm.12}$
RandReg-$\mathcal{N}(\mu_s, \sigma_s^2)$	$72.08^{\pm.50}$	$80.20^{\pm.01}$	$90.61^{\pm.24}$	$71.77^{\pm.78}$	$95.16^{\pm.15}$	$91.81^{\pm.11}$
RandReg-CP	$72.10^{\pm.20}$	$80.02^{\pm.20}$	$90.55^{\pm.17}$	$72.34^{\pm.27}$	$95.86^{\pm.30}$	$91.76^{\pm.61}$
AdaRand (Ours)	$\mathbf{74.60}^{\pm.10}$	$\mathbf{81.27}^{\pm.26}$	$\mathbf{91.17}^{\pm.13}$	$\mathbf{74.86}^{\pm.22}$	$\mathbf{96.68}^{\pm.14}$	$92.34^{\pm.26}$

GPU with 40GB VRAM and recorded average test accuracy with standard deviation evaluated on the final models.

6.2. Evaluation on Multiple Pre-trained Models

One of our motivations is to develop a method that achieves high performance on arbitrary pre-training methods without relying on auxiliary source information and additional heavy computation. Here, to verify that this motivational goal is achieved, we evaluate AdaRand with multiple combinations of pre-training methods and architectures such as CLIP and ViT-B. In this experiment, we compare our method with RandReg, FNP, and DR-Tune, which are available without auxiliary source information. Table 2 shows that AdaRand stably outperforms RandReg and the other baselines for all combinations of pre-training methods and architectures. In particular, AdaRand overcomes the negative effects on SimCLR observed in Sec. 4. These results indicate that the adaptive prior update of AdaRand is widely effective for many pre-trained models.

6.3. Evaluation on Multiple Datasets

We evaluate AdaRand on multiple different datasets to demonstrate its generality across the datasets. Table 3 shows the results. Except for the Pets dataset, AdaRand achieved the best results for all of the datasets. While the performance of RandReg depends on the hyperparameters of priors, AdaRand stably performs by the adaptive prior update.

6.4. Evaluation on Small Datasets

We evaluate AdaRand by reducing the data volume of Cars into $\{10, 25, 50\}\%$. Table 4 shows that AdaRand still outperforms the baselines even with a limited dataset of less than a few training samples per class (i.e., 10%). The results demonstrate that AdaRand performs well in data-scarce scenarios.

Table 4. Top-1 test accuracy (%) on small training datasets (Cars, ResNet-50 pre-trained with ImageNet classification).

Method / Dataset Size (%)	10 %	25 %	50%
Fine-tuning	$19.58^{\pm.07}$	$53.10^{\pm.08}$	$77.86^{\pm.09}$
FNP [10]	$23.68^{\pm.34}$	$57.07^{\pm.34}$	$79.93^{\pm.21}$
DR-Tune [38]	$23.68^{\pm.34}$	$57.07^{\pm.34}$	$79.93^{\pm.21}$
RandReg-$U(0,1)$	$25.35^{\pm.14}$	$58.33^{\pm.45}$	$80.91^{\pm.51}$
RandReg-$\mathcal{N}(0,1)$	$26.15^{\pm.21}$	$59.94^{\pm.58}$	$81.74^{\pm.41}$
RandReg-$\mathcal{N}(\mu_0, \sigma_0^2)$	$24.95^{\pm.05}$	$59.66^{\pm.24}$	$81.31^{\pm.17}$
RandReg-CP	$26.45^{\pm.26}$	$59.37^{\pm.46}$	$81.13^{\pm.23}$
AdaRand (Ours)	$\mathbf{27.55}^{\pm.10}$	$\mathbf{61.09}^{\pm.54}$	$\mathbf{82.19}^{\pm.19}$

Table 5. Comparison among fine-tuning regularization methods (Cars, ResNet-50 pre-trained with ImageNet classification). We measure top-1 test accuracy, averaged training time per epoch, and GPU memory usage. Our method (AdaRand) outperforms the other baselines in accuracy while maintaining reasonable computation time and GPU memory consumption.

Method	Test Accuracy (%)	Time / Epoch (sec.)	GPU Mem. (MiB)
Fine-tuning	$88.29^{\pm.12}$	**9.737**	**8,073**
FNP [10]	$90.27^{\pm.06}$	9.916	8,073
L2SP [18]	$88.50^{\pm.25}$	12.405	8,153
DELTA [19]	$89.00^{\pm.24}$	14.733	9,211
BSS [5]	$89.70^{\pm.06}$	11.403	8,227
Core-tuning [36]	$90.01^{\pm.13}$	16.872	24,223
DR-Tune [38]	$90.38^{\pm.59}$	25.475	8,845
Co-Tuning [33]	$90.66^{\pm.34}$	11.546	8,125
UOT [21]	$90.82^{\pm.19}$	12.405	14,293
RandReg-$U(0,1)$	$90.42^{\pm.24}$	9.988	8,075
RandReg-$\mathcal{N}(0,1)$	$90.32^{\pm.41}$	9.971	8,075
RandReg-$\mathcal{N}(\mu_s, \sigma_s^2)$	$90.61^{\pm.24}$	9.963	8,075
RandReg-CP	$90.55^{\pm.17}$	11.552	8,075
AdaRand w/o ℓ_{intra}	$90.81^{\pm.21}$	12.410	8,585
AdaRand w/o ℓ_{inter}	$90.99^{\pm.06}$	12.841	8,585
AdaRand (Ours)	$\mathbf{91.17}^{\pm.13}$	13.824	8,585

6.5. Evaluation on Comparison to SoTA Methods

We demonstrate the efficacy of AdaRand by comparing it with the SoTA baselines. Table 5 shows the results on the Cars dataset with the ResNet-50 architecture pre-trained

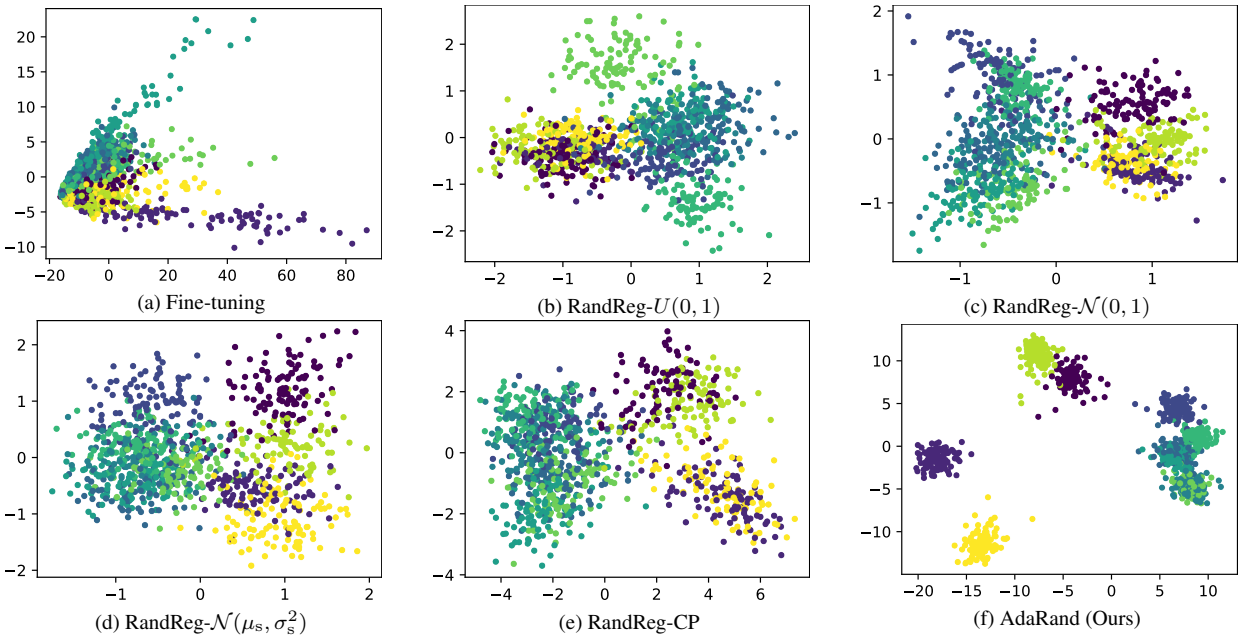


Figure 3. PCA visualization of feature spaces of trained models (CIFAR-10, ResNet-50). The colors in the sample plot correspond to that class. AdaRand clearly forms well-separated clusters, which can be useful for solving downstream classification tasks.

on the ImageNet classification task. We confirm that our AdaRand achieved better performance than the methods leveraging auxiliary source information (i.e., Co-Tuning and UOT) or additional heavy computation costs (i.e., Core-tuning and DR-Tune). The bottom part of Table 5 also shows an ablation study of AdaRand, where RandReg-CP is a method using the conditional prior defined in Eq. (7) without updating them. AdaRand significantly improved the performance of RandReg-CP and the losses of ℓ_{inter} and ℓ_{intra} complementarily contributed to the improvements. This indicates that naïvely introducing the conditional prior is not sufficient to solve the challenges of RandReg and adaptively updating priors is effective in terms of performance.

6.6. Analysis and Discussion

Through the experiments in the previous sections, we validate that AdaRand stably improves the accuracy of the downstream classification tasks in various settings and overcomes the RandReg's challenge caused by using fixed priors discussed in the former paragraph of Sec. 4.3. Here, we provide the analysis of AdaRand to confirm whether it overcomes the rest challenges of RandReg, i.e., decreasing feature norms $\|g_\phi(x)\|_2^2$ and entropy $H(g_\phi(x))$. Figure 2 demonstrates that AdaRand succeeds in preventing the decreasing $\|g_\phi(x)\|_2^2$ and $H(g_\phi(x))$. As a result, AdaRand achieves better the gradient norm of cross-entropy loss $\|\nabla_W \ell_{\text{CE}}(f_\theta(x), y)\|_2^2$ and mutual information $I(g_\phi(x); y)$ than RandReg. This also can be an explanation for the reason why AdaRand stably outperforms RandReg in the previous sections.

Finally, we discuss the effects of AdaRand on feature spaces by visualizing feature vectors from trained models with the PCA dimension reduction. As the dataset, we used CIFAR-10 [16]. We trained ResNet-50 by each method and reduced the dimensions of the output of $g_\phi(x)$ by PCA. We randomly selected 1,024 samples from the test set for the input. The visualization results are illustrated in Figure 3. While the baseline method forms less separated feature clusters, AdaRand clearly forms independent and dense feature clusters for each class. This indicates that the adaptive prior update of AdaRand helps models learn useful representations for solving downstream classification tasks.

7. Conclusion

This paper presents a novel regularization method for fine-tuning deep neural networks called AdaRand. AdaRand penalizes feature vectors of deep models by guiding to the random reference vectors that are generated from the class conditional prior distributions. To encourage the fine-tuning models to generate useful features for target classification tasks, AdaRand adaptively updates the conditional prior distributions so that the prior distributions are close to the current feature distribution and balance the distance between classes. Through this simple method, The fine-tuned model increases the amount of mutual information between features and labels, resulting in a significant improvement in final test accuracy without either auxiliary source information or additional heavy computation costs. An important future step is to extend AdaRand beyond discriminative tasks such as generative modeling by diffusion models.

References

[1] Pulkit Agrawal, Ross Girshick, and Jitendra Malik. Analyzing the performance of multilayer neural networks for object recognition. In *European conference on computer vision*, 2014. 3

[2] Christopher M Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995. 5

[3] Malik Boudiaf, Jérôme Rony, Imtiaz Masud Ziko, Eric Granger, Marco Pedersoli, Pablo Piantanida, and Ismail Ben Ayed. A unifying mutual information view of metric learning: cross-entropy vs. pairwise losses. In *European conference on computer vision*, pages 548–564, 2020. 4

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 2020. 3, 4, 6, 7

[5] Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. In *Advances in Neural Information Processing Systems*, 2019. 2, 6, 7

[6] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2014. 6

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 6

[8] Lev Faivishevsky and Jacob Goldberger. Ica based on a smooth estimation of the differential entropy. In *Advances in neural information processing systems*, 2008. 4

[9] Weifeng Ge and Yizhou Yu. Borrowing treasures from the wealthy: Deep transfer learning through selective joint fine-tuning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1086–1095, 2017. 2

[10] Bharath Hariharan and Ross Girshick. Low-shot visual recognition by shrinking and hallucinating features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3018–3027, 2017. 1, 3, 4, 6, 7

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016. 3, 6

[12] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *International Conference on Computer Vision*, pages 4918–4927, 2019. 1

[13] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 3

[14] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations*, 2019. 4

[15] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition*, Sydney, Australia, 2013. 3, 6

[16] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 8

[17] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in neural information processing systems*, 2018. 5

[18] Xuhong Li, Yves Grandvalet, and Franck Davoine. Explicit inductive bias for transfer learning with convolutional networks. In *International Conference on Machine Learning*, 2018. 1, 2, 6, 7

[19] Xingjian Li, Haoyi Xiong, Hanchao Wang, Yuxuan Rao, Liping Liu, Zeyu Chen, and Jun Huan. Delta: Deep learning transfer using feature map with attention for convolutional networks. In *International Conference on Learning Representations*, 2019. 1, 2, 6, 7

[20] Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Xiangyang Ji, Antoni Chan, and Rong Jin. Improved fine-tuning by better leveraging pre-training data. 2022. 1

[21] Ziquan Liu, Yi Xu, Yuanhong Xu, Qi Qian, Hao Li, Xiangyang Ji, Antoni B. Chan, and Rong Jin. Improved fine-tuning by better leveraging pre-training data. In *Advances in Neural Information Processing Systems*, 2022. 2, 6, 7

[22] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 6

[23] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. *arXiv*, 2013. 6

[24] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008. 6

[25] Tianyu Pang, Chao Du, and Jun Zhu. Max-mahalanobis linear discriminant analysis networks. In *International Conference on Machine Learning*, pages 4016–4025. PMLR, 2018. 3

[26] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2012. 6

[27] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 2021. 1, 3, 4, 6, 7

[28] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3), 2015. 1

[29] Masaaki Takada and Hironori Fujisawa. Transfer learning via ℓ_1 regularization. In *Advances in Neural Information Processing Systems*, 2020. 3

[30] Weitao Wan, Cheng Yu, Jiansheng Chen, Tong Wu, Yuanyi Zhong, and Ming-Hsuan Yang. Shaping deep feature space towards gaussian mixture for visual classification. *IEEE transactions on pattern analysis and machine intelligence*, 45 (2):2430–2444, 2022. 3

[31] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010. 6

[32] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, 2014. 3, 6

[33] Kaichao You, Zhi Kou, Mingsheng Long, and Jianmin Wang. Co-tuning for transfer learning. *Advances in Neural Information Processing Systems*, 2020. 1, 2, 6, 7

[34] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, 2021. 3, 4, 6, 7

[35] Shihao Zhang, Linlin Yang, Michael Bi Mi, Xiaoxu Zheng, and Angela Yao. Improving deep regression with ordinal entropy, 2023. 4

[36] Yifan Zhang, Bryan Hooi, Dapeng Hu, Jian Liang, and Jiashi Feng. Unleashing the power of contrastive self-supervised visual models via contrast-regularized fine-tuning. *Advances in Neural Information Processing Systems*, 2021. 1, 6, 7

[37] Yang Zhong and Atsuto Maki. Regularizing cnn transfer learning with randomised regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13637–13646, 2020. 1, 3, 4, 6

[38] Nan Zhou, Jiaxin Chen, and Di Huang. Dr-tune: Improving fine-tuning of pretrained visual models by distribution regularization with semantic calibration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 1, 6, 7